ATARI

# ST COMPUTER

Die Fachzeitschrift für den ATARI-ST Anwender.

März 90
DM 7,-
Ös. 56,-
Sfr. 7,-
3

## Turbo C 2.0
## & Debugger

## 650 MB
**Magnetooptisches Laufwerk**

## GFA- BASIC 3.5

MMAD

# Erste, zweite, dritte und vierte Hilfe.




*Die vier sind zwar nicht von der Tankstelle, aber als Werkzeuge im täglichen ST-Leben – wir möchten sagen – so gut wie unentbehrlich.*

Unter dem Decknamen *Utility Series* haben wir vor einiger Zeit eine neue Reihe von kleinen aber feinen Hilfsprogrammen angekündigt, bei denen der Teufel im Detail steckt. Heute gibt es vier solcher Tools und ein feinsinniger Beobachter vermag nun auch eine Reihe zu sehen: ❶ *FlexDisk,* die Ramdisk für den blitzschnellen Zugriff auf alle Daten machte den Anfang. Ihre Größe läßt sich ganz nach Bedarf einstellen, man kann Accessories und Autoordner von einem beliebigen Laufwerk starten, damit ist immer das nötige Werkzeug zur Stelle. ❷ Sicherheitskopien von der Harddisk macht man ohne Umstände mit dem *Harddisk Utility.* Bei beliebiger Dateigröße sichert man auch sehr große Datenbestände auf Diskette. Durch Rückspeichern auf eine beliebige Harddisk oder Ramdisk können Dateien mit diesem Programm auch von einem Massenspeicher auf einen anderen transferiert werden. Das Harddisk-Backup-Programm ist einfach, sicher und schnell zu bedienen und natürlich GEM eingebunden. ❸ *Protos* ist relativ neu auf dem Markt, aber schon jetzt zeichnet sich seine Beliebtheit deutlich ab, weil es sozusagen das Schweizer Taschenmesser unter den Utilities ist. *Protos,* das aus dem Autoordner gestartet wird und speicherresident bleibt, bietet eine Reihe nützlicher Funktionen, die das Arbeiten nicht nur mit dem ST-Bildschirm angenehm komfortabel werden lassen. Einzelheiten entnimmt der Interessent aus dem Software-Info *Protos,* das wir auf Anfrage gerne zusenden. ❹ Schließlich: *Disk Utility,* das Routinen rund um Diskette und Harddisk enthält und als Accessory, als Programm und unter *Signum!* benutzt werden kann. Aus verschiedenen Programmen aufrufbar, unterhält es so umfangreiche Features, daß wir auch hier auf das entsprechende Software-Info verweisen wollen. Ansonsten: Hals- und Beinbruch!

Die Kunst der ST-Software.

III
APPLICATION
SYSTEMS
HEIDELBERG

**Application Systems Heidelberg** Software GmbH, Englerstraße 3, Postfach 10 26 46, D-6900 Heidelberg 1, Telefon (0 62 21) 30 00 02, Fax (0 62 21) 30 03 89 **in Österreich:** Reinhart Temmel Ges.m.b.H., Markt 109, A-5440 Golling, Telefon (0 62 44) 7081-0, Fax 7 18 83. **in der Schweiz:** DTZ DataTrade AG, Landstraße 1, CH-5415 Rieden/Baden, Telefon (0 56) 82 18 80, Fax 82 18 84.

# EDITORIAL

Die Vorurteile gegenüber dem ATARI ST scheinen kein Ende zu nehmen. Gerade im Lager vieler PC-Benutzer entstehen immer wieder neue Gerüchte, die man sich als ST-Anwender nur schwer anhören kann, ohne den Kopf zu schütteln. "Desktop-Publishing ist mit so einem Spiel-Computer nicht möglich, der Ventura Publisher ist das Nonplusultra. Gibt es überhaupt ein DTP-Programm für den ST?" fragt mein Gegenüber, eingefleischter MS-DOS-Anwender seit einem Jahr. Mit einer kurzen Demonstration von Calamus beweise ich das Gegenteil. Wie es denn mit einer Textverarbeitung aussehe, fragt mein Gegenüber. Eine kurze Demonstration von Script und Wordplus läßt den verdutzten MS-DOSler kurzzeitig verstummen. Die nächste Antwort, die ich ihm gebe, bringt ihn ganz aus dem Konzept, denn er fragt, mit welchem Laserdrucker der Textausdruck gemacht worden sei, der auf meinen Schreibtisch liegt. Belustigt zeige ich auf meinen 24-Nadeldrucker und lade Signum! ein. Nach ungefähr einer Minute wortlosen Herumstehens und Staunens eilt mein Gegenüber aus der Redaktion. Warum er so schnell weglief, konnte ich nicht mehr fragen.

## MS-DOS versus TOS

Ein anderes Beispiel: In Frankfurt ist momentan überall Werbung der Sparkasse für "Electronic Banking" zu sehen. Interessiert rief ich meine Sparkasse an, um dort enttäuscht zu erfahren, daß ein Datenträgeraustausch mit dem ATARI ST nicht möglich ist, weil bisher noch kein Programm zur Verfügung stehe, mit dem dergleichen möglich sei. Ob der Spiel-Computer überhaupt für solche Zwecke geeignet sei, fragt mich der freundliche Herr. Ich habe den freundlichen Herrn überzeugen können und in die Redaktion eingeladen. Das Ergebnis des Gesprächs steht schon jetzt fest: Das Programm, das die Sparkasse benötigt, wurde von mir bereits entwickelt und ihr zur Verfügung gestellt. So wird der ATARI ST auch Einzug in den formularlosen Datenträgeraustausch halten.

Anhand dieser Beispiele können Sie leicht sehen, daß das Bild des ATARI ST immer noch nicht das Bild des echten ATARI ST ist. ATARI hat immer noch das Image eines Spiel-Computer-Herstellers, was auch irgendwo verständlich ist - aber das ist ein anderes Thema und soll an anderer Stelle besprochen werden.

Sollten Sie also von einem "Fremd-Betriebssystem-Benutzer" auf den ST angesprochen werden, klären Sie am besten Unklarheiten auf. Nur so kann das Image des doch recht professionellen Geräts ATARI ST aufgebessert werden. Was ATARI selbst nicht schafft, muß eben von den Benutzern in die Hand genommen werden. Ich habe mir jedenfalls schon einen Standard-Antwortenkatalog gegenüber Vorurteilen von MS-DOS-Benutzern zurechtgelegt, der auch in den meisten Fällen anwendbar ist.

Martin Pittelkow

*Liebe Leser und Leserinnen,*

leider sehen wir uns gezwungen, den Verkaufspreis der ST Computer ab der nächsten Ausgabe auf DM 8,- zu erhöhen. Wenn Sie mal ein bißchen in Ihren alten Ausgaben nachblättern, werden Sie feststellen, daß dies die erste Preiserhöhung seit November 1987 ist. Sie sehen also, daß wir solange wie möglich den Preis gehalten haben, obwohl ständig an der ST Computer verbessert wird und auch ihr Umfang seit damals deutlich zugenommen hat.

*Vielen Dank für Ihr Verständnis*

# INHALT

## SOFTWARE

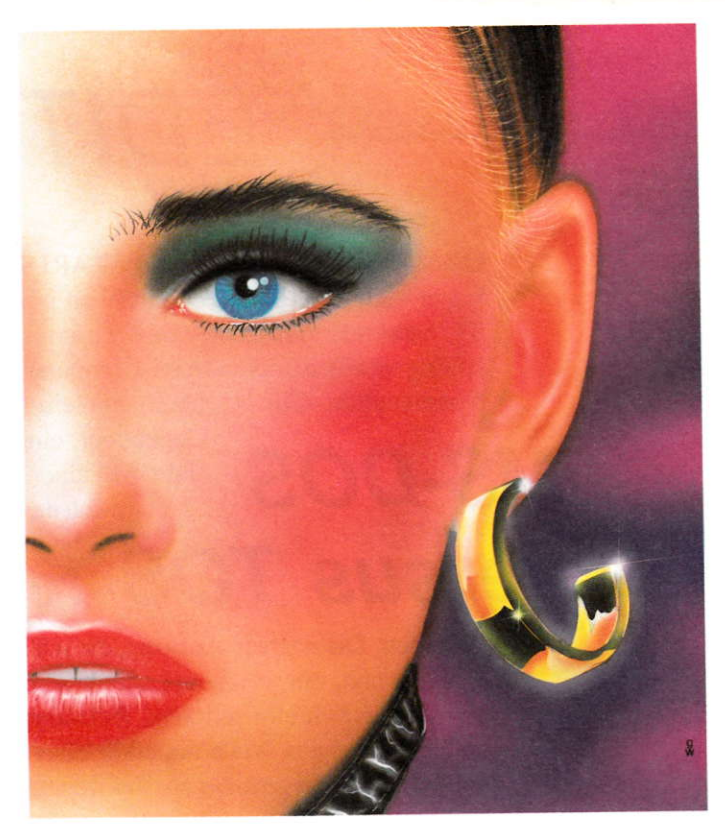


## ANWENDUNGEN



## HARDWARE

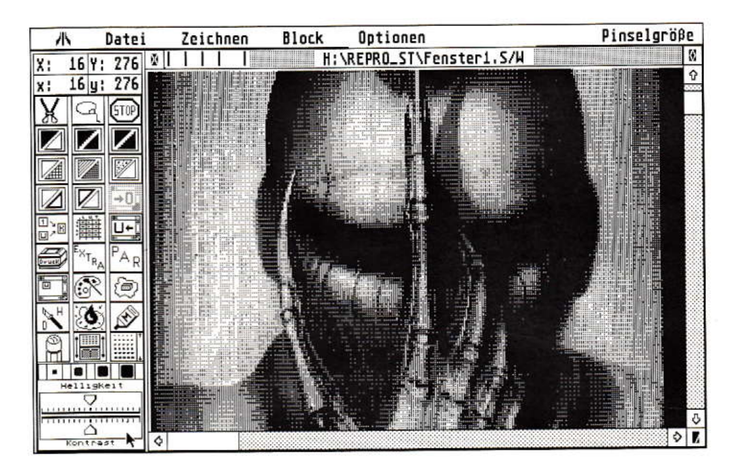




## Turbo C 2.0 und sein Source Level Debugger

Als Borland vor einigen Monaten auf dem ST-Markt in Erscheinung trat, waren viele überrascht. Nicht Turbo Pascal, das immer wieder angekündigt worden war und bis heute nicht erschienen ist, war nun zu erwerben, sondern Turbo C wurde Borlands erstes Zugpferd. Die Kunden stürzten sich geradezu auf den neuen mit ANSI-Standard ausgerüsteten C-Compiler. Jetzt bringt Borland mit Turbo 2.0 seine neue Version auf den Markt, die als Unterstützung ebenfalls einen Source-Level-Debugger zur Seite bekommt.

Seite **49**




## Repro Studio ST

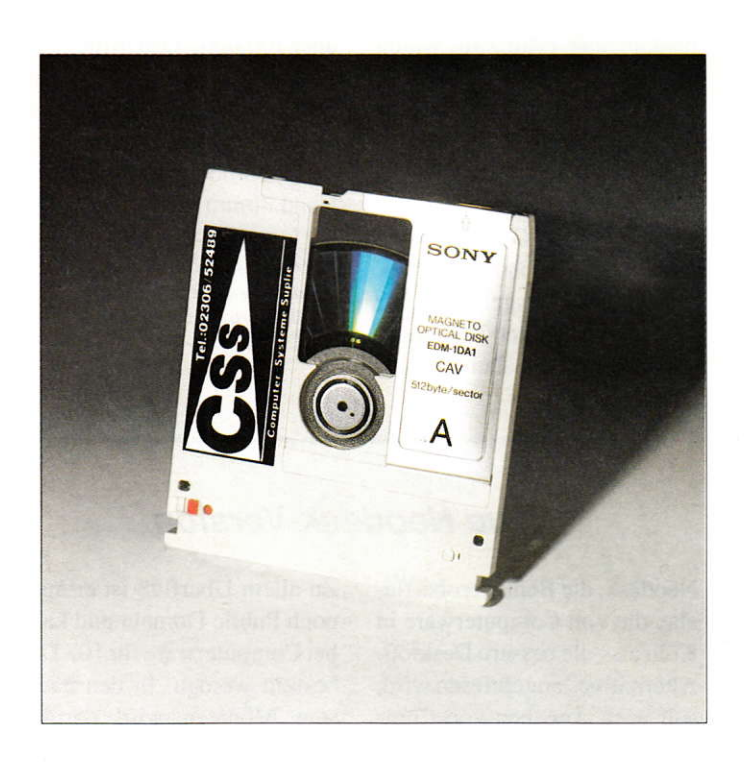
Gemäß der eigenen Aussage der Entwickler ist Repro Studio ST hauptsächlich dafür gedacht, die Bearbeitung digitalisierter Bilder zu erleichtern. Jene digitalisierten Bilder erhält der Anwender entweder über Scanner oder über Videodigitizer, und genau in diese Zielgruppe paßt sich Repro Studio ST ein.

Seite **12**




Die meisten Computer-User wissen im allgemeinen sehr wenig über das Innenleben eines Tastaturprozessors. Dieser dreiteilige Artikel soll deshalb die Neugier derer stillen, die schon immer mehr über das unbekannte Gebiet der vielen Tasten wissen wollten, und den knappen Informationsstand der restlichen Leser auf Vordermann bringen, die lieber die vielen Anwenderprogramme auf dem Markt des ATARI ST zu Rate ziehen, als sich selber mit der Programmierung zu beschäftigen.

Seite **126**




## Optischer Riese - CSS Gigafile 650

Mal mußte es so kommen: Kaum haben wir uns - ehemals noch stolz auf glatte 720 kB Diskettenkapazität - an die Megabytes auf den Festplatten gewöhnt, wollen uns alerte Nachswuchskonzerne mit gigantischen Datenschluckern beglücken. Genauer: Mit einem optomagnetischen Laufwerk, das 650 Meckerbytes beherbergt. Es kommt von der jungen Firma CSS und nennt sich GIGAFILE 650.

Seite **54**

## GRUNDLAGEN



## PROGRAMMIERPRAXIS



## AKTUELLES



## RUBRIKEN

# NEWS

## Turbo ST 1.8
## Noch schneller, noch leistungsfähiger

Turbo-ST liegt nun in der Version 1.8 vor. Gegenüber der alten Version 1.6 hat sich einiges getan. Die bekannten Fehler (z.B. mit Signum2!) sind behoben worden, und die bisherigen Funktionen sind natürlich auch wieder um ein paar Prozentpunkte schneller. Wirklich neu ist, daß jetzt auch diagonale Linien und Polygone unterstützt werden, wovon zum Beispiel vektororientierte Programme wie CAD-Anwendungen besonders profitieren können. Außerdem ist das Auffrischen des Bildschirmes spürbar schneller, was sich im Desktop und allen Programmen bemerkbar macht, die das GEM-Interface benutzen. Pixelorientierte Malprogramme gewinnen durch die neue Polygon-Füll- sowie die verbesserte Raster-Copy-Routine deutlich an Tempo. Ferner wird jetzt auch der 6x6-Systemzeichensatz und farbiger Text unterstützt. Wer sich den Accessory-Eintrag sparen will, der kann ab sofort die Auto-Ordner-Version verwenden, die mitgeliefert wird. Speziell für Programmierer gedacht ist die Möglichkeit, Turbo ST von einem Programm aus- und einschalten zu können. Auch die Anwender eines Matrix-Großbildschirmes werden nicht enttäuscht. Für den 19"-Monitor mit einer Auflösung von 1280x960 liegt ebenfalls eine Version vor. Der Preis für die neue Turbo ST Version 1.8 wird voraussichtlich DM 89,- (ohne Gewähr) sein. Alle Besitzer der alten Version 1.6 können für DM 30,- zu einem Update gelangen.

*BELA COMPUTER*
*Unterortstrasse 23-25*
*6236 Eschborn*
*Tel.: 06196/481944*

## ATARI UltraScript

Nach langer Wartezeit wird jetzt endlich ATARIs Post-Script-Interpreter UltraScript ausgeliefert. Ein Testbericht über die Verkaufsversion war bereits in der ST 5/89 zu lesen. Neu ist, daß die Version mittels drei verschiedener RSC-Dateien dreisprachig (französisch, englisch, deutsch) ausgeliefert wird. Aus unbekannten Gründen hatte sich die Auslieferung verzögert. Zusätzlich sind für insgesamt DM 248,- drei zusätzliche Fonts erhältlich, die nach Adobe-Maßen erstellt sind. Dabei handelt es sich um die Schriftfamilien Times, Helvetica und Courier. Damit sollte die gängigsten Anwendungen abgedeckt sein.

*Alle ATARI-Fachhändler*

## AdiDOC
## Dokumentenverwaltungssystem

Für die Verwaltung unstrukturierter Daten, z.B. Notizen, Produktbeschreibungen, Zeitschriften- und Buchauszüge, stellt die Karlsruher Firma ADI Software auf der diesjährigen CeBIT AdiDOC vor. Richter am BGH Karlsruhe, deren komplexe Textdaten und schwierigste Recherche-Anforderungen perfekte Datenbanktechnik erfordern, waren seit 1989 maßgeblich beteiligt, dem Dokumentenverwaltungssystem AdiDOC Marktreife zu geben. AdiDOC wird nach der CeBIT für die Betriebssysteme TOS (DM 399,-), DOS (DM 795,-) und UNIX (DM 2850,-) ausgeliefert. Für Anbieter "gefüllter Textdatenbanken" werden Runtime-Lizenzen angeboten.

Auf Initiative der RA Consulting, Graben-Neudorf, unterstützt die Karlsruher ADI Software den DDR-Wahlkampf mit ihrem Datenbanksystem Adimens. Dadurch können auch kleinere demokratische Gruppen im Umfeld des Neuen Forums ihre Organisation verbessern. Anlaufstelle für die ersten Pakete, die im Wert von über 100.000,- DM in die DDR gesendet wurden, ist das Neue Forum in Jena, das die Vergabe der Software an nichtextreme Gruppen überparteilich in die Hand nimmt.

*ADI Software GmbH*
*Bunsenstraße 22*
*7500 Karlsruhe*
*Tel. 0721/8203-0*

## Neue Neodesk-Version

Neodesk, die Benutzeroberfläche, die von Computerware in Köln als "die bessere Desktop-Alternative" angepriesen wird, soll nach Angaben von Computerware noch besser werden. Jetzt gibt es in Neodesk einen "rettenden Papierkorb", aus dem man das einmal Gelöschte ohne Probleme wieder hervorholen kann. Der "rettende Papierkorb" kann ab der Version 2.05 eingesetzt werden. Er benutzt die neue Kommunikationsschnittstelle von Neodesk und wird so vollkommen in die Arbeitsumgebung integriert. Zu allem Überfluß ist er auch noch Public Domain und kann bei Computerware für 10,- DM bestellt werden. In den nächsten Monaten wird darüber hinaus ein "Neodesk-Entwickler-Paket" zu haben sein, das es dem Anwender ermöglicht, sein eigenes Neodesk-Zubehör zu entwickeln, das diese Schnittstelle zu seinem Vorteil benutzt.

*Computerware*
*Gerd Sender*
*Weißer Straße 76*
*5000 Köln 50*
*Tel. 0221/396186*

| | |
|---|---|
| Garamond 10 Pt. Roman | ABCDEFGHIJKLMNO<br>abcdefghijklmnopqrstuvwyx |
| Garamond 11 Pt. Roman | ABCDEFGHIJKLMN<br>abcdefghijklmnopqrstuv |
| Garamond 12 Pt. Book | ABCDEFGHIJKLM<br>abcdefghijklmnopqrs |
| Garamond 13 Pt. Light Condensed | ABCDEFGHIJKLMN<br>abcdefghijklmnopqrst |

Application Systems /// Heidelberg eröffnet mit Garamond eine neue Schriftenreihe. Wie im Bild zu sehen ist, handelt es sich dabei um hochwertige Schriften, die über den bisherigen Qualitätsmerkmalen angesiedelt sind. Garamond ist bisher nur in 24-Nadlerversion zu haben, die Laserversion der Schrift ist bereits in Arbeit. Der Preis beträgt DM 100,-. Bei Application Systems ist jetzt auch die Bodoni (neue Version) von Semiotic Soft aus München für DM 199,- und die Schönschrift von Herrn Schön, ebenfalls aus München, zu haben. Schönschriften gibt es für 24-Nadler und Laser (jeweils DM 199,- oder beide zusammen für DM 299,-). Signum! selbst wird jetzt auf drei Disketten ausgeliefert. Auf der dritten Diskette befinden sich alle Standard-Fonts in der Laserversion, der Druckertreiber und Font-Editor für Laserdrukker und ein Konvertierungsprogramm für 24-Nadel-Fonts auf Laser-Fonts. Damit wird die Signum!-Printerdisk überflüssig.

Das Entwicklungssystem Laser C wird von Application Systems ab sofort nur noch als integriertes Komplettsystem verkauft. Dieses Paket enthält den C-Compiler mit Linker, eine grafische Shell, einen schnellen Texteditor, ein Resource-Construction-Programm, viele Unix-Kommandos und als besonderes Bonbon den Quelltext-Debugger LaserDB. Der Compiler verfügt über einen eingebauten Inline-Assembler und ist laut Application Systems derzeit der wohl schnellste verfügbare C-Compiler auf dem ATARI ST. Die schnellen Mathe-Software-Routinen benutzen jetzt automatisch den Arithmetik-Coprozessor, sofern einer im Rechner eingebaut ist. Das System ist auf allen ST-TOS-Verisonen inklusive TOS 1.4 lauffähig. Mit diesem Komplettangebot wird ein Entwicklungssystem angeboten, das allen Interessenten einen günstigen Einstieg in die Programmiersprache C ermöglichen soll. Alles zusammen kostet DM 348,-.

*Application Systems /// Heidelberg*
*Englerstraße 3*
*6900 Heidelberg*
*Tel. 06221/300002*

## Terminverwaltung

Mit DATE-ACCESS kann nun jeder ATARI ST-Besitzer eine Terminverwaltung als Accessory auf seinem Computer installieren. Damit haben auch vergeßliche Anwender die Möglichkeit, alle Termine komfortabel auf einen Blick zu haben. DATE-ACCESS kostet DM 49,- (ohne Versandkosten) und ist zu beziehen über

*Jensen Software*
*Dorotheenstraße 45*
*2390 Flensburg*
*Tel. 0461/57815*

## GMa-Soft-News

Das Programm ST-Fibu der Firma GMa-Soft aus Offenbach wurde weitreichend verbessert. Buchungen von externen Programmen können jetzt eingelesen und verarbeitet werden. Der Kontenrahmen ist jetzt frei erstellbar, ein automatischer Jahresübertrag ist auch zu einem späteren Zeitpunkt (z.B. erst im März) möglich. Wichtige Funktionen können jetzt durch ein Code-Wort vor unberechtigter Benutzung geschützt werden.

Mit ST-Fakt bietet GMa-Soft ein Programm an, das die Adressendatei der ST-Fibu nutzen kann und beim Rechnungsschreiben gleich die Buchungen für die ST-Fibu erstellt. ST-Fakt kann bis zu 2000 Artikel/Leistungen verwalten. Auf eine Lagerverwaltung wurde bewußt verzichtet, da das Programm in erster Linie für kleinere und mittlere Handwerks- und Dienstleistungsbetriebe gedacht ist.

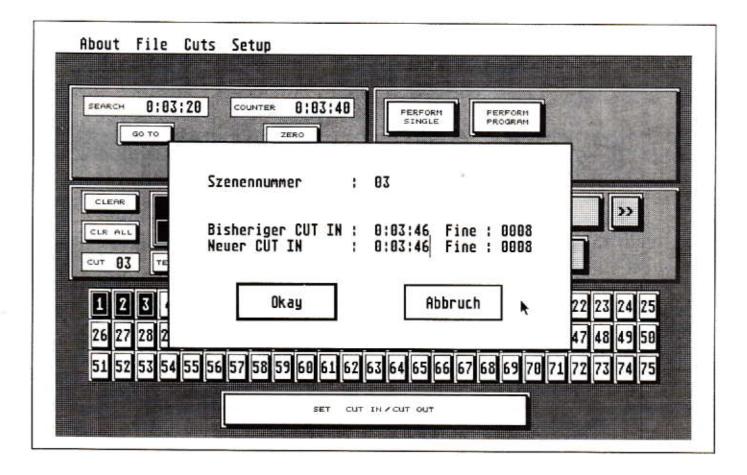
ST-GMa-Text ist ein Textsystem, das sich in zwei Bereiche gliedert: eine schnelle Textverarbeitung mit Serienbrieffunktion und ein Mahnwesen zum automatischen Erstellen von Mahnungen. Sowohl die Serienbrieffunktion als auch das Mahnwesen nutzen die Daten der ST-Fibu. Durch den Zugriff auf die Adreßdaten der ST-Fibu können leicht Serienbriefe erstellt werden. ST-Fibu kostet DM 398,-. Für das Zusatzmodul 1 mit Mahnwesen, Textverarbeitung und Serienbrief müssen DM 150,- bezahlt werden, das Zusatzmodul 2 mit Fakturierung und Übernahme in die Fibu kostet DM 200,-.

*GMa-Soft*
*Bergstraße 18*
*6050 Offenbach*
*Tel. 069/898345*

## VideoEd 8

VideoEd 8 heißt das neue Programm für Video 8-Filmer. Dieses Soft- und Hardware-Paket ermöglicht dem Videofilmer eine Schnittbearbeitung von 75 Szenen mit einem ATARI ST. Während der Video-Wiedergabe merkt sich das Programm die Anfangs- und Schlußpositionen einer Szene per Mausklick. Die Szenen können mit einem Text versehen und ediert werden. Ist die Edierung am Computer beendet, wird der Videofilm per "Knopfdruck" geschnitten. Das Programm kann an verschiedene Video 8-Kameras und Aufnahmerekorder angepaßt werden. VideoEd 8 kostet SFr. 298,-.

*CD-Mailing*
*Uetlibergstraße 33*
*CH-8045 Zürich*
*Tel. 0041/01/4613883*



## PC-Speed für STE

Der erfolgreiche Hardware-Emulator PC-Speed ist jetzt auch für den ATARI STE fertig. Für diesen ATARI-Rechner wurde eine völlig neue Platine entwickelt, die speziell die Fähigkeiten des STE unterstützt. Durch den Einsatz von SMD-Technik und einen neuen Steckadapter findet PC-Speed in diesem Computer problemlos Platz, ohne daß man löten muß. Die Auslieferung beginnt ab dem 20. Februar über alle ATARI-Fachhändler.

*Sack Electronic GmbH*
*Bleichstraße 49*
*4792 Bad Lippspringe*
*Tel. 05252/4290*

## Der Rubel rollt!

Die Kunden des Unternehmens TommySoftware nehmen die Werbung des Software-Hauses beim Wort. Derzeit läuft eine Werbekampagne mit dem Titel "Glasnost die Erste", die beispielsweise in der ST-Computer bis Februar abgedruckt wurde. Damit wird das Grafikprogramm Mega-Paint II umworben. Was ursprünglich als augenzwinkernder Scherz gedacht war, entpuppte sich als Flachs mit überraschendem Effekt: TommySoftware forderte in den Anzeigen die Kunden auf, die Probediskette in DM oder den Gegenwert in sowjetischen Rubeln zu begleichen. Prompt haben einige Kunden sich die Mühe gemacht und die Software-Probe mit Rubeln geordert. Mittlerweile hat die sowjetische Regierung ihre Währung stark abgewertet. Die vielen zugesandten Rubel sind faktisch nur noch wenige Pfennige wert.

*TommySoftware*
*Selchower Straße 32*
*1000 Berlin 44*
*Tel. 030/621406-3*

## DARC-Fragenprogramm

Seit der Interradio 1989 wird das Programm "Software-Paket Fragenkatalog" zur Ausbildung und Vorbereitung für die Amateurfunklizenz über den Landesjugendverband Niedersachsen des DARC (Deutscher Amateur-Radio-Club) vertrieben. Diese Datensammlung enthielt zunächst nur den Text des "Fragenkatalogs" der Deutschen Bundespost für den C64 und diente dazu, sich Fragebögen zur Prüfungsvorbereitung zusammenzustellen. Dieses Paket ist nun auch für alle ATARI ST-Rechner erhältlich. An einer Aufbereitung von Grafiken wird momentan noch gearbeitet.

*Deutscher Amateur-Radio-Club*
*Landesjugendverb. Niedersachsen*
*Herbert Frager, DF1AW*
*Kapellenberg 26*
*3411 Kaltlenburg*

```
ECH.NIK\INHALT.TXT
B e t r i e b l i c h e   K e n n t n i s s e

1.2   Abwicklung des Amateurfunkverkehrs
1.3   Buchstabiertafel
1.4   RST-System
1.5   Q-Schluessel, soweit fuer den Amateurfunk erforderlich
1.6   Amateurfunkabkuerzungen
1.7   Die wichtigsten Landeskenner fuer den Amateurfunk
1.8   Stationstagebuch und QSL-Karten
```

## dBMAN mit deutschem Handbuch

Zu dBMAN V gibt es endlich das deutsche Handbuch. Es enthält auch die Beschreibung von Kommandos, die im Original nicht zu finden sind (SELFILE, SEND, RECEIVE). Ein Beispielprogramm im Anhang des Handbuchs soll den Einstieg erleichtern, ein deutscher Tutor und Assist geben weitere Hilfestellungen dazu. Außerdem hat der Benutzer jetzt die Wahl zwischen englischen und deutschen Online-Hilfstexten. Zusammen mit dem deutschen Handbuch kostet das Update von Version 5.0 auf 5.2 DM 179,-. Nichtregistrierte User können diesen Service der Firma Mai zum Preis von DM 398,- in Anspruch nehmen.

*Computer Mai*
*Metzstraße 19*
*8000 München 80*
*Tel. 089/4480691*

## Neue MagicBox ST-Version

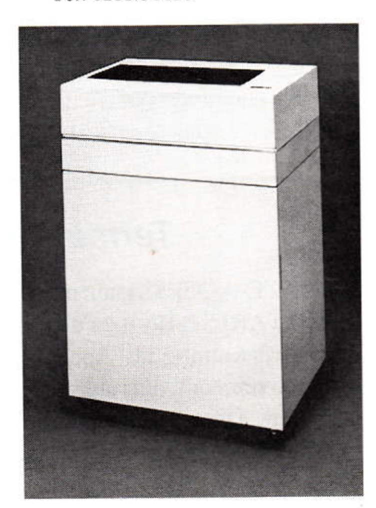
MagicBox ST, das Mailbox-Programm zum Anschluß an das MagicNET, ist jetzt in der Version 7.71b erhältlich. Die neue Programmversion beherrscht jetzt außer X- und ZModem auch XModem 1K, YModem und YModem Batch. Die Option YModem G für das MNP 5-Protokoll ist bereits eingebaut, wird aber noch nicht unterstützt. Alle Käufer der Sonderdisk-Version, die ein Update haben möchten, schicken am besten einen mit DM 2,40 frankierten, an sich selbst adressierten Rückumschlag an PECES oder rufen in der PEC-Mailbox an und bekommen das Programm dort per Download. Der Preis der Vollversion hat sich gesenkt; MagicBox ST mit Anschluß an das MagicNET kostet jetzt nur noch DM 149,-.

*PECES*
*Tucholskystraße 41*
*6000 Frankfurt 70*
*Tel. 069/683584 (Mailbox)*

## High Speed-Zeilendrucker

Der neue Hochleistungsdrucker CI-1000 von C.ITOH ist als Matrix-Zeilendrucker mit schwingender Hammerbank konzipiert. Er beinhaltet vier programmierbare Menüeinstellungen für vier verschiedene Benutzer. Wer viel auszudrucken hat, wird begeistert sein von der Geschwindikeit des Geräts: bei 10 Zeichen pro Inch schiebt der CI-1000 das Papier mit einer Geschwindigkeit von 933 Zeilen pro Minute in den Auswurfkorb. In LQ-Qualität druckt er immerhin noch 200 Zeilen pro Minute. Wer diese Geschwindigkeit nicht missen möchte, muß für das Gerät DM 21.660,- zahlen (ohne Unterschrank). Der CI-1000 ist ab April erhältlich.

*C.ITOH Eelectronics GmbH*
*Immermann-Hof*
*Immermannstraße 65d*
*4000 Düsseldorf 1*
*Tel. 0211/36850*



## 150 MB-Streamer

Seit einiger Zeit steht ein professioneller Streamer für den ATARI ST zur Verfügung. Damit sind schnelle Backups von Festplatten kein Problem mehr. Der Streamer sichert innerhalb einer Minute 6,5 MB Daten. Die zugehörige Software läßt unter TOS und DOS ein Backup und Restore ganzer Partitionen sowie einzelner Dateien zu. Das Gerät verfügt über eine SCSI-Schnittstelle und einen durchgeschleiften DMA-Bus, außerdem über eine batteriegepufferte Echtzeituhr. Der anschlußfertige Streamer kostet inklusive Software DM 2498,-.

*Weide Elektronik GmbH*
*Regerstraße 34*
*4010 Hilden*
*Tel. 02103/41226*

## NEC richtet bundesweit Geschäftsstellen ein

Die Geschäftsleitung der NEC Deutschland GmbH hat beschlossen, in Zukunft die NEC-Produkte Drucker und Monitore nicht nur über die Großhändler, sondern auch über den Fachhandel anzubieten. "Zu diesem Zweck haben wir Anfang Januar weitere sechs dezentrale Geschäftsstellen eröffnet." Mit dieser Entscheidung bekräftigt Masao Takahashi, Managing Director der NEC Deutschland GmbH, die Absicht des Unternehmens, den direkten Kundenkontakt auszubauen. NEC-Geschäftsstellen befinden sich nun in Berlin, Düsseldorf, Frankfurt, Hamburg, Hannover, München und Stuttgart. Über 150 neue Stellen wurden zur Verwirklichung dieses zusätzlichen Dienstleistungsangebots im Fiskaljahr 1989, das am 31.3.1990 endet, geplant.

*NEC Deutschland GmbH*
*Klausenberger 4*
*8000 München 80*
*Tel. 089/937776/8*

## Autoboot-Programm XBoot

Ein neuartiges Boot-Programm namens XBoot soll den Komfort aller bisherigen Boot-Lader überbieten. XBoot läßt sich komplett per Maus bedienen, speichert häufig benötigte Einstellungen als SET ab, kann die Reihenfolge der zu ladenden Programme im Auto-Ordner beliebig verändern, sortiert die Anzeige von PRGen und ACCs usw. Bei Rechnern ohne akkugepufferte Uhr bleiben Uhrzeit und Datum auch nach einem Reset erhalten und vieles mehr. XBoot ist ferner konfigurierbar über ein separates Programm. XBoot kostet DM 49,- und ist zu beziehen über

*TN Software*
*Tassilo Nitz*
*Krablerstraße 24*
*4300 Essen 12*

## SGSNet - Midi-Netzwerk

SGSNet ist ein einfaches Midi-Netzwerk, geignet zur Verbindung von 32 ATARI ST-Computern mit einer maximalen Verbindungslänge von 150m. Es ermöglicht den Zugriff auf alle Festplattenpartitionen oder RAM-Disks des File-Servers. Alle Netzwerkfunktionen laufen im Hintergrund und werden über die mitgelieferte Netzwerk-Software gesteuert. Datenübertragungsfehler werden erkannt und automatisch korrigiert. Ebenfalls sind Zugangsberechtigungen und Benutzerbeschränkungen vom Fileserver aus einstellbar. SGSNet kostet DM 448,- für zwei ST-Rechner, für jeden weiteren müssen DM 298,- bezahlt werden.

*Makro C.D.E.*
*Schillerring 19*
*8751 Großwallstadt*
*Tel. 06022/24405*

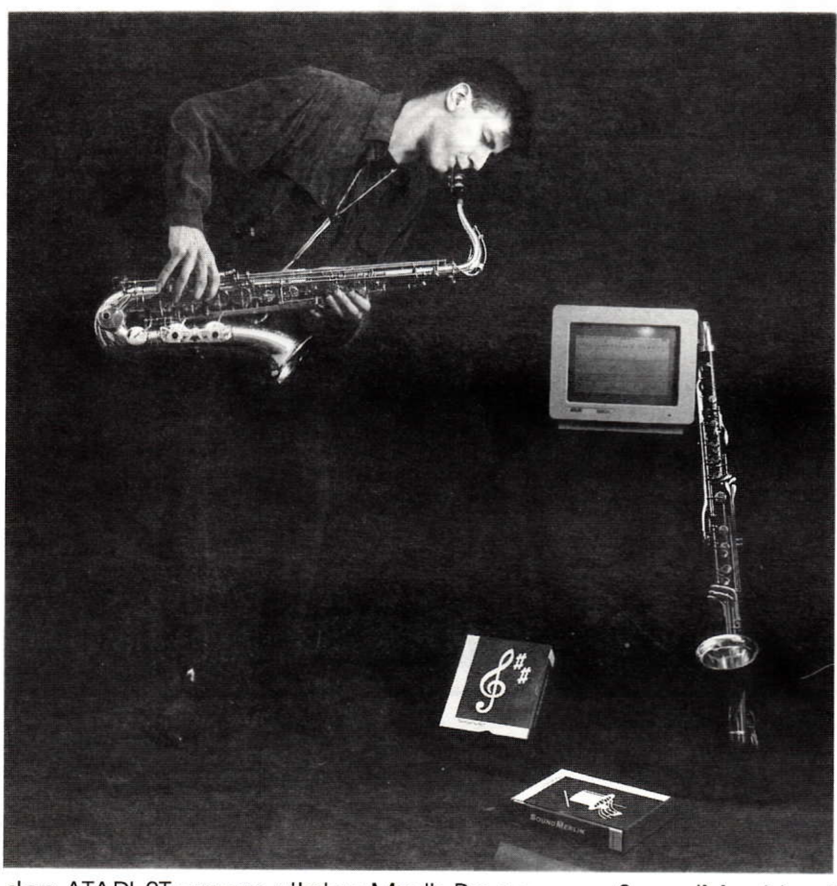
## Fehlerteufelchen

In der letzten Ausgabe stellten wir WordPerfect 4.1 vor. Leider ist die Bezugsadresse nicht korrekt, WordPerfect 4.1 ist zu beziehen über alle ATARI-Fachhändler, nicht, wie angegeben, über WordPerfect in Eschborn.

Der Bericht über Mortimer enthielt einige Unklarheiten, die hiermit berichtigt werden sollen. Uns stand zum Test aus Versehen eine Vorabversion zur Verfügung. Folgendes ist in der Endversion enthalten: Mit der Lupe lassen sich alle Funktionen ausführen, die auch im normalen Betrieb des Rechners durchgeführt werden können. Die RAM-Disk läßt sich abspeichern und nach dem nächsten Starten von Mortimer automatisch wieder einladen. Mortimer funktioniert jetzt auch mit Tempus. Das einzige Programm, das bekanntermaßen nicht mit Mortimer läuft, ist Protext.

TOMMYSOFTWARE®
Überlegen durch Kreativität

# TONKÜNSTLER UNTER SICH.



Musik ist für viele Menschen ein faszinierendes Erlebnis. Doch nur die wenigsten haben die Möglichkeit, ein Instrument zu erlernen. Und für alle, die trotzdem ihre eigene Musik kreieren wollen, empfehlen wir für den ATARI-ST unsere vitalen Musik-Programme SoundMachine II und SoundMerlin.

SoundMachine II ist das Universaltalent. Zur Grundausstattung gehören 74 wirklichkeitsgetreue Instrumente, von denen mehrere gleichzeitig aufspielen können. Ob Sie in die Trompete blasen, in die Saiten der Baß-Gitarre greifen oder Keyboards spielen – SoundMachine II bietet Klangqualitäten, die echtem Synthesizer-Sound entsprechen.

SoundMerlin ist das peppige Sample-Editor-Programm mit bis zu 100 Samples im Speicher. Es integriert verschiedene Module wie Sequenzer, Drumcomputer oder Keyboard. Mit vielfältigen Manipulationsmöglichkeiten, z.B. Ein-/Ausblenden, Mischen, Kopieren, Ausschneiden und Transponieren, geben Sie Ihren Samples den letzten Schliff. SoundMerlin ist kompatibel zu allen gängigen Samplern, und ist auch für den Einsatz mit MIDI–Keyboards vorbereitet.

Mit SoundMachine II und SoundMerlin bringen Sie Ihre Musik in Form.

Probedisketten und ausführliche Informationen erhalten Sie gegen Einsendung von DM 20,–.

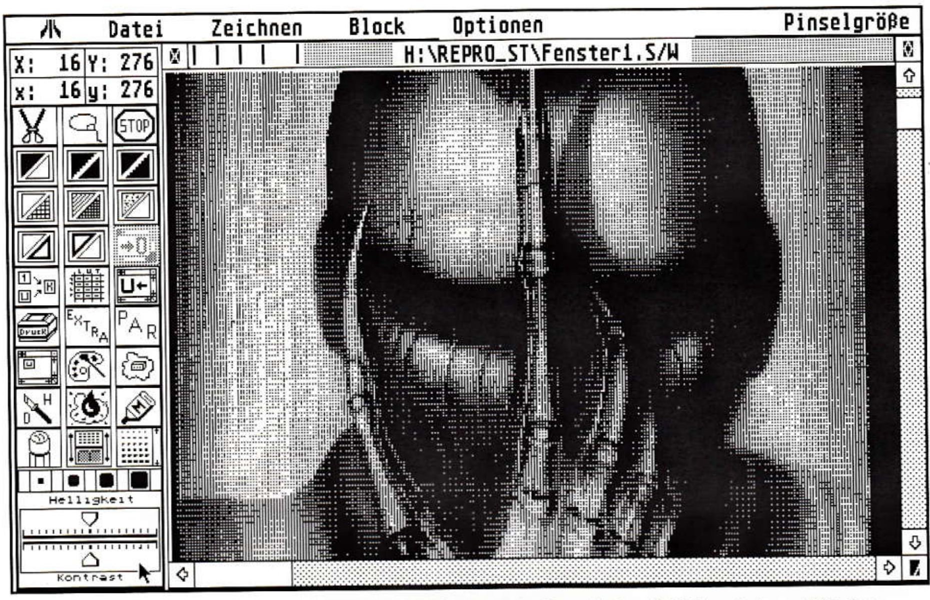
C 8912131 -pb-

TOMMYSOFTWARE®
Selchower Str. 32 c
D-1000 Berlin 44
Tel. 030/ 621 40 6-3
Fax/BTX 030/ 621 40 6-4

# Die Bildwerkstatt

# Repro Studio ST

**Grafikprogramme gibt es wie Sand am Meer. Ob es nur darum geht, mal kurz eine kleine Zeichnung zu erstellen, oder (ungleich komplizierter und umfangreicher) eine Trickfilmsequenz zu programmieren - Grafikprogramme gibt es für alle (möglichen und unmöglichen) Fälle.**




*Das Hauptarbeitsfenster von Repro Studio ST mit der Icon-Leiste "Bildbearbeitung" (links)*

Seit einiger Zeit wird das Angebot an Scannern, jener 'Bildeinlesegeräte', etwas reichhaltiger, und oft sind dort auch schon kleinere Bildbearbeitungsprogramme beigelegt. Lohnt es sich dann noch, ein separates Bildmanipulationsprogramm zu kaufen? Um die Antwort vorwegzunehmen (laut Radio Eriwan): Im Prinzip JA!

Gemäß der eigenen Aussage der Entwickler ist Repro Studio ST hauptsächlich dafür gedacht, die Bearbeitung digitalisierter Bilder zu erleichtern. Jene digitalisierten Bilder erhält der Anwender entweder über Scanner oder über Videodigitizer, und genau in diese Zielgruppe paßt sich Repro Studio ST ein.

## Allerlei Eingabegerät

Die Digitalisierer, welche bislang für den ATARI ST auf dem Markt erhältlich waren, orientierten sich immer an der Bildschirmgröße. Will heißen: Es konnten immer nur Bilder eingelesen werden, die in der Größe der Bildschirmauflösung entsprachen (bei dem SM 124 Schwarzweißbildschirm sind das 640 x 400 Punkte - genannt Pixel). Digitalisierer neuerer Bauart bieten da schon höhere Auflösung, etwa: 1024 x 512 Pixel.

Bei den Scannern sieht es noch schlimmer aus. Eine DIN-A4-Seite bei 200 DPI (engl. Abk. f.: Dots per Inch, Punkte pro Zoll) beansprucht 1700 x 2200 = 3.740.000 Pixel. Der ATARI Schwarzweißmonitor realisiert aber nur 640 x 400 = 256.000 Pixel, das sind nicht einmal 15% der Vorlagenfläche (DIN-A4).

Repro Studio ST löst dieses Problem durch einen 'virtuellen Bildspeicher', in dem die komplette Vorlage festgehalten wird. So sind alle Zeichen-, Verschiebe- oder Veränderungsfunktionen nicht auf den sichtbaren Bereich beschränkt. Dieser Trick macht das Arbeitsfeld nur noch von der RAM-Speicherkapazität abhängig, damit funktioniert Repro Studio ST (logischerweise) auch mit einem Groß-/Ganzseitenbildschirm.

## Lasset uns beginnen

Mit einem Doppelklick auf *REPRO.PRG* starten wir und harren der Dinge, die da kommen: AHA! Eine Warnbox meldet, daß der Druckertreiber fehlt - und den Scanner-Treiber habe ich auch vergessen. Aber das ist nicht weiter schlimm, denn alle Treiber sind jederzeit im Programm nach- bzw. umladbar (besonders interessant für "fliegenden Wechsel", während das Programm noch arbeitet).

Es erscheint anschließend die Arbeitsoberfläche, die oben eine Desktop-Menüleiste und links eine Icon-Leiste zeigt, der große Rest des Bildschirms bleibt (noch) leer.

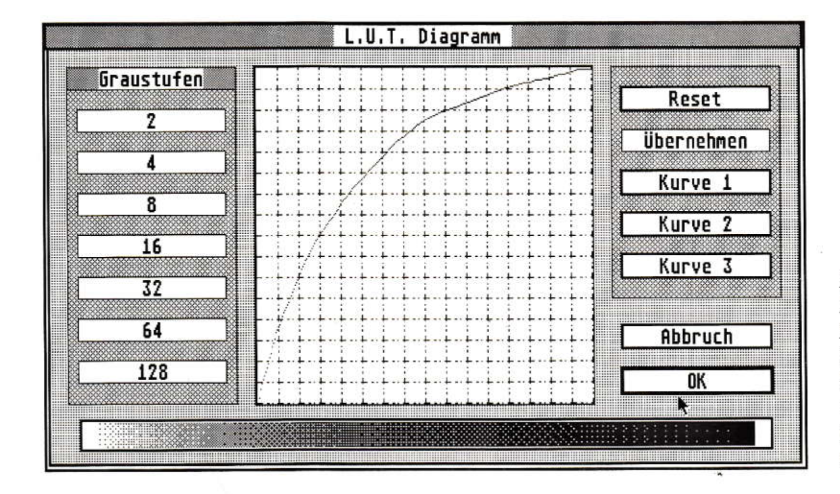
Über eine eigene Fileselektorbox *Bild laden* soll nun ein schon fertig vorhandenes Bild auf die Arbeitsoberfläche geholt werden. Es sind dabei zwei Alternativen zu beachten: Entweder ist 1. schon ein (leeres) Arbeitsfenster geöffnet, dann legt sich das zu ladende Bild in die Größeneinstellung dieses Fensters (Pauschalvorgabe - mit der Gefahr, daß das Fenster zu klein ist), oder 2. wird das Bild ohne vorher offenes Fenster gerufen. Dabei bestimmt das Bild mit seinen Abmessungen die Größe des Fensters (Individualvorgabe).

Apropos: Es sind bis maximal 4 geöffnete Fenster möglich, was sehr stark von deren Größe im Verhältnis zum freien RAM-Arbeitsspeicher abhängig ist. Die Auswahl an lad- und speicherbaren Bildformaten ist groß.

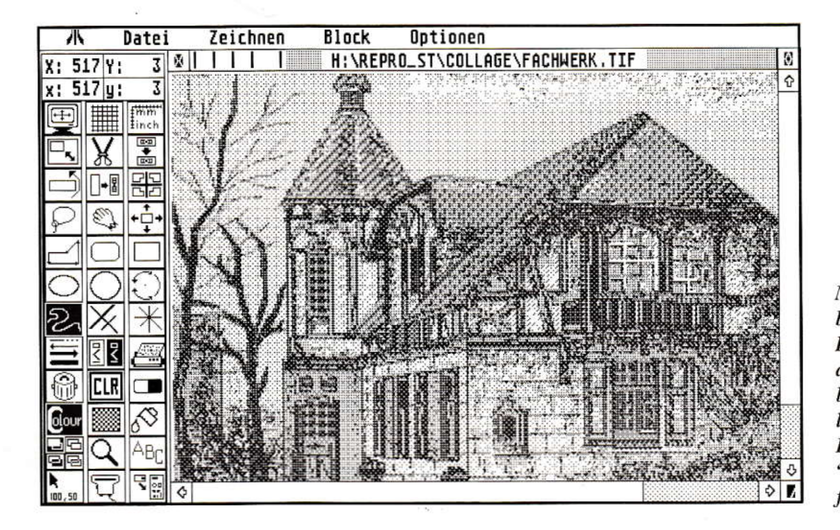
Repro Studio ST versteht folgende Normen: DEGAS (auch gepackt), STAD, Paintworks, IMG, TIF, MPK und Art Director; in Vorbereitung: IFF (AMIGA) und PCX (MS-DOS).

## Farblos oder bunt

Repro Studio ST unterscheidet nicht nur alleine für die Darstellung auf einem entsprechenden Monitor zwischen schwarzweiß und farbig. Viele Funktionen in der Bildbearbeitung sind sinnvollerweise nur in der passenden "Colorierung" der Vorlage anwählbar.




*In dem L.U.T.-Diagramm ist der Helligkeitsverlauf jeder Graustufe einstellbar.*




*Neben einem bearbeiteten Bild zeigt das Hauptarbeitsfenster links die Icon-Leiste "Zeichenfunktionen".*

| Datei | Zeichnen | Block | Optionen |
|---|---|---|---|
| Fenster öffnen | Ausschnitt invert | Bild Y spiegeln ^Y | Funk. ausweiten ^A |
| Bild laden ^L | Parallelogramm ^J | Bild X spiegeln ^X | Funk. gefüllt ^G |
| Bild in Puffer | Raute ^M | Bild verkleinern ^K | UNDO ausweiten ^U |
| speichern ^S | Vieleck ^T | Bild 90° drehen ^D | Bild nachrücken ^N |
| speichern als | Kreisviertel ^F | Bild invertieren ^I | Maskieren ^C |
| Bildformate | Bezierkurven ^B | Bild zoomen ^Z | Überblenden ^R |
| Diskoperationen | Sprühdose ^E | Block invertieren | Bildbearbeitung ^H |
| Druckertreiber | Voreinstellung ^P | Block verzerren | Bereich schützen |
| ext. Module ^ ESC | | Block um 90° drehen | load Lasso / Maske |
| Programmende ^Q | | Halbton stauchen | save Lasso / Maske |

*Die Pull-Down-Menüs von Repro Studio ST; fast alle Funktionen sind auch per Control-Sequenz ansteuerbar.*

*1. Schwarzweiß oder Rasterbilder (Grauton)*

- Glätten: In ungeraden Linien kommen oft Treppensprünge vor, die nicht sehr angenehm aussehen. Diese Unebenheiten können einander angepaßt werden, sogar mehrmals für dasselbe Bild.
- Kanten: Großflächige (gleichfarbige) Bereiche, z.B. Großbuchstaben, kann man hiermit auf ihre Umrandung beschränken. Der Inhalt der Fläche verschwindet, es bleibt nur noch eine Umrißlinie zurück. Anschließend wären Fülleffekte (z. B. mit Raster oder Mustern) möglich.
- Verwaschen: Scharfe Kontrastgegensätze im Bild (direkter Übergang von Schwarz auf Weiß) sind mit diesem "Weichzeichner" etwas sanfter zu gestalten. Die Übergänge werden fließender.
- Grauwerte filtern: Der Filter entfernt alle Grauwerte, die zwischen zwei einstellbaren Schwellwerten liegen. Damit lassen sich genau definierbare Konturen (mit demselben Grauwert) herauslöschen, die z.B. durch Überbelichtung oder Reflexion entstanden sind.
- Grauwerte oder Muster tauschen
- Wandlung in Halbton- oder Zufallsrasterbild
- Aufhellen und verdunkeln, global oder nur auf einzelne Bildabschnitte (Blöcke) bezogen
- Kanten verstärken oder ausdünnen
- Zoomen bis 600% der Ausgangsgröße
- L.U.T.-Diagramm: Es bestimmt für jeden möglichen Grauwert eine frei wählbare Helligkeitsstufe. Der Kurvenverlauf dieses Diagramms macht die Manipulation sehr anschaulich.

Als sehr interessant muß (last but not least) die Wandlung von Schwarzweißbildern (Raster) in Halbtonbilder (pseudofarbig) erwähnt werden. Was sich damit alles "colorieren" läßt... Auch sind stufenlose Helligkeits- und Kontraständerungen mit den dadurch erzielbaren Effekten als zusätzliche Features ansteuerbar.

*2. Farb- bzw. Halbtonbilder*

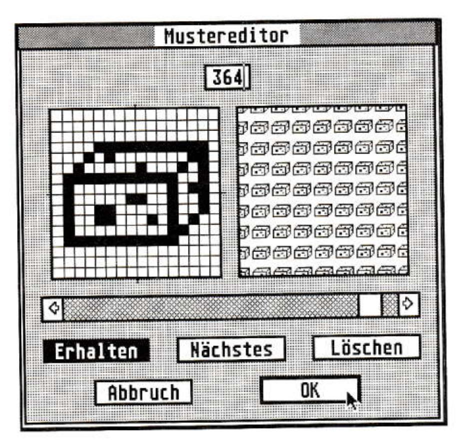
Neben zahlreichen ähnlichen Funktionen aus dem Schwarzweißbereich, wären noch folgende erwähnenswert:

- Verlaufsraster erzeugen
- Aufrauhen mit Zufallsgenerator
- Zoomen bis 1600% der Originalgröße

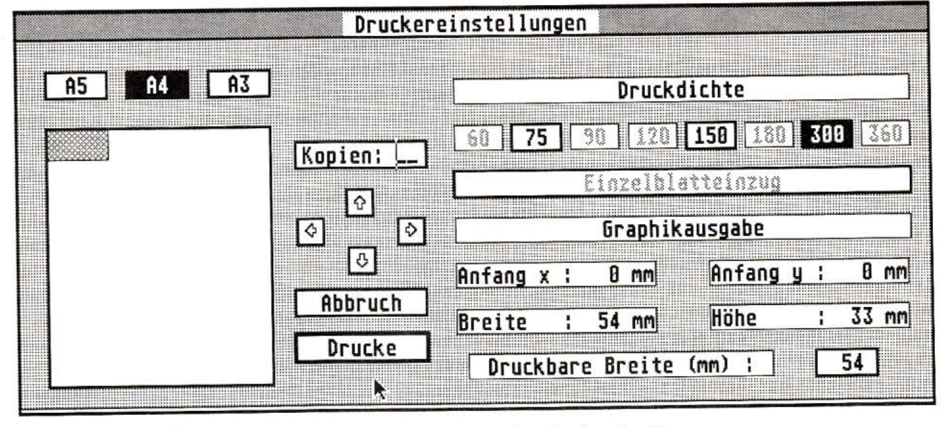
Alle Funktionen können zusätzlich in ihrer Wirkung auf nur eine oder mehrere von 256 RGB-Farbstufen beschränkt werden.

## Zeichenfunktionen

Die Werkzeuge im Zeichenteil von Repro Studio ST sind nun nicht mehr abhängig davon, ob ein Schwarzweiß- oder Farbbild vorliegt. Weil auch hier die Palette sehr reichhaltig ist, wollen wir uns auf die Betrachtung einiger wesentlicher Funktionen beschränken:




*Ein Mustereditor erlaubt die Erzeugung eigener Hintergrundraster.*




*Viele Einstellungen lassen die freie Plazierung der Vorlage im Ausdruck zu.*

- Scrollen: Wenn mit dem Fadenkreuz ein Bildrand erreicht wird, läuft das verdeckte Bildteil automatisch in das Fenster nach. Es muß also nicht ein GEM-üblicher Rollbalken bewegt werden.
- Objekte ausschneiden, verschieben, kopieren, verzerren, drehen, spiegeln, invertrieren, feinjustieren
- Beziérlinien, Füllmuster, Vollbildeinblendung, Lupenfunktion
- Zeichnen von Linie, Rechteck, Raute, Kreis, Kreisbogen, Kreisabschnitt usw.

Bei den reinen Zeichenoperationen darf man feststellen, daß kaum noch Wünsche offenbleiben.

## Mit vielerlei Format

Derzeit sind folgende Scanner- und Digitizer-Ansteuerungen verwirklicht: Hawk CP-14, Print Technik Professional, A-MAGIC Turbo Dizer, Geniscan; in Vorbereitung sind: Siemens, Epson, Sharp, SPAT, Panasonic.

Wenn es Nutzer des Programms gibt, die einen Treiber für etwas unübliche oder seltene Scanner-/Digitizer-Typen gerne in Repro Studio ST eingebunden wünschen, möge man sich gerne mit der Firma Hofmann in Verbindung setzen. Die Entwickler sind gerne bereit, spezielle Ansteuerungen sofort in Arbeit zu nehmen. Druckertreiber gibt es für: fast alle 24-Nadler, ATARI Laser SLM 804, HP-Laserjet-Kompatible.

## Was bringt die Zukunft?

Selbst bei dieser Fülle an Möglichkeiten bleibt die Entwicklung nicht stehen. So sind noch zahlreiche Erweiterungen geplant:

- Vektorisierungsmodul zur Erzeugung von Objektformat
- integrierter Texteditor, auch für Vektorschriften (Metafile)
- mathematisch ermittelbare Verlaufsraster
- frei gestaltbare Rastergrößen und -winkel
- Anpassung an die MAXON-MGE-Farbgrafikkarte.

## Der Versuch eines Überblicks

Es war überraschend, wieviele Funktionen in Repro Studio ST zu finden sind. Und ich bitte gleich um Entschuldigung, daß es mir nicht möglich ist, hier alle erschöpfend zu beschreiben. Ich glaube sogar, daß vielleicht einige Spielarten des Programms von mir erst gar nicht entdeckt wurden. Die Auswahl ist riesengroß! Sehr angenehm ist die Tatsache, daß wirklich nur sehr selten benutzte Menüpunkte in den Pull-Downs untergebracht und (im Gegensatz dazu) die häufig gebrauchten in den zwei Icon-Leisten zu finden sind.

Repro Studio ST, speziell für die Manipulation digitalisierter Bilder entwickelt, läßt dem Benutzer, ob Hobby- oder Profianwender, sicherlich kaum noch Wünsche offen - mir jedenfalls sind nach mehr als 8 Stunden (ehrlich!) intensiver Beschäftigung mit dem Programm keine mehr eingefallen. Außerdem kann sich die Geschwindigkeit im Bildbearbeitungsteil sehen lassen. Mangels Vergleich mit Konkurrenzprodukten ist diese Einschätzung natürlich subjektiv.

Als sehr schön ist auch das Handbuch zu werten, in dem alle Programmteile ausführlich und mit genügend Bildmaterial

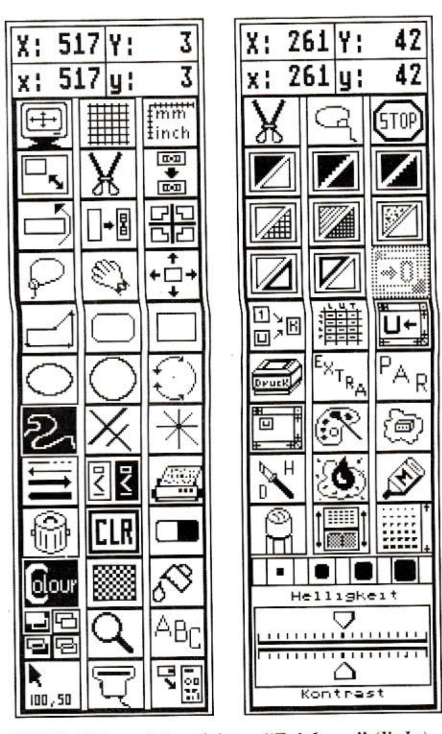



*Bild 7: Die zwei Iconleisten "Zeichnen" (links) und "Bildbearbeitung" (rechts)*

erläutert sind. Der Anhang erklärt außerdem an reichlich bebilderten Beispielen verschiedene Tricks der Bildmanipulation.

Als Kundendienst bietet die Firma Hofmann Software engineering selbstverständlich Update- und Hotline-Betreuung an. Im Preis von 498,- DM sind alle Scanner- und Druckeransteuerungen enthalten. Mögliche neue oder geänderte Treiber (wenn sich die Gerätekonfiguration des Kunden ändert) werden kostenlos nachgeliefert.

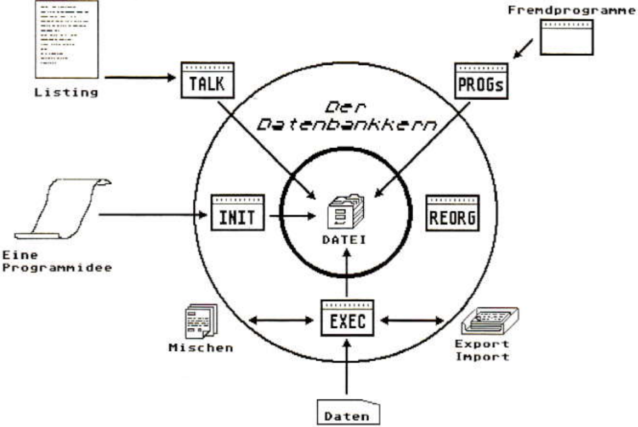
*DK*

***Bezugsadresse:***

*Bernd Hofmann Software engineering*
*Lilienweg 12*
*6834 Ketsch*
*Tel.: 06202/61393*

# AdiTALK ST plus
# Die nächste Generation

**Unlängst durften wir die neueste Version des Datenbankprogramms Adimens ST plus begutachten und schnell feststellen, daß sich dort bemerkenswerte Neuerungen eingestellt haben. So war es nicht verwunderlich, daß AdiTALK als kleineres Mitglied dieser Produktfamilie ziemlich bald auch einen "Frühjahrsputz" erfahren konnte.**



*Die Adimens-Familie*

AdiTALK stellt eine Datenbankabfragesprache, ähnlich einer Programmiersprache, dar, mit der sich ständig wiederkehrende Arbeitsabläufe automatisierbar machen lassen. Durch den Einsatz von AdiTALK verzichtet der Benutzer zwar auf Desktop, Masken, Menüs und Maus (wie sie in "Adimens" beispielsweise benutzt werden), erhält aber dadurch mehr Geschwindigkeit bei der eigentlichen Datenarbeit, dem Eingeben, Suchen, Sortieren und Ausgeben. Übrigens: Alle Adimens-typischen Tätigkeiten lassen sich in ein AdiTALK-Programm einbauen. Zum Lieferumfang von AdiTALK ST plus gehören neben dem Hauptprogramm *TALK* und vier kleineren Entwickler-Tools zwei Programme, die es auch bei Adimens gibt: dies wären das Datenbankvereinbarungsmodul *INIT* und das Reorganisationsmodul *REORG*. (Über die beiden Letztgenannten möchten wir heute keine Worte mehr verlieren - ist bereits geschehen im Heft 11/89 dieser Zeitschrift.) Es ist durchaus denkbar, daß ein Anwender nicht den Kauf-"Umweg" über Adimens machen möchte. Dann bekommt er im AdiTALK-Paket alle Werkzeuge geliefert, die zum Aufbau und Betreiben einer Datenbank nötig sind.

## Was ist neu?

### 1. Verbunde

Im Vereinbarungsteil *INIT* legt der Benutzer die spätere Struktur der Datenbank fest. Dies geschieht noch in gewohnter GEM-Manier mit Maus, Menüs und Symbolen. Für die spätere Arbeit in *TALK* steht auch das neue Merkmal "Verbunde" zur Verfügung. In der *VIEW*-Funktion wird eine eigens erzeugte Verbundmaske benutzt, um nur bestimmte Datenbankfelder anzeigen zu lassen. Bei der *JOIN*-Funktion ist es sogar möglich, einzelne Felder aus verschiedenen Dateien in einer Maske zusammenzulegen und zu bearbeiten.

### 2. Neues REORG

*REORG* wird benutzt, wenn in einer Datenbankdatei Felder hinzugefügt, verändert oder weggenommen werden sollen. Dadurch ändern sich die Struktur und die Anordnung der Datensätze.Neben einer gewaltigen Geschwindigkeitsverbesserung besitzt dieses Abgleichprogramm nun auch eine eigene GEM-Oberfläche.

### 3. Einbinden von Druckerdateien

Was bisher von zahlreichen Anwendern schmerzlich vermißt wurde, ist nun endlich verwirklicht worden: eigene Druckertreiber für AdiTALK. Damit sind andere Zeichensätze erreichbar geworden, und die Druckertreiber können während des Programmablaufes gewechselt werden.

Standardmäßig greift AdiTALK für einen Ausdruck immer auf die Druckerkonfigurationsdatei *TALK.CFG* zu. Jetzt wurde der Befehl *set printer to* für die freie Anwahl anderer Druckerdateien (*.CFG) erweitert:

```
SET PRINTER TO (Ausgabe) (<Name>.CFG),
Zeilenlänge, Seitenlänge
```

### 4. Das Zeichensatz-Accessory

Als kleines Bonbon an die MS-DOS-verwöhnten Anwender auf dem ATARI ST liegt ein Zeichensatzwandler als Accessory bei. Jetzt können die hebräischen Zeichen beispielsweise gegen IBM-Grafiksymbole getauscht werden. Interessant wird diese Funktion z.B. in einer statistischen Berechnung, um daraus Balkengrafik zu erzeugen. Anderen Realisationen sind eigentlich dann keine Grenzen mehr gesetzt.

### 5. Das Modul TALKER

Für AdiTALK- Anwendungsprogramme werden oft Menüs sowie Ein- und Ausgabemasken gebraucht. Hier übernimmt *TALKER* die eigentliche Programmierarbeit.

Beispiel: Wir möchten eine Maske erzeugen. Zunächst genügt ein einfacher Texteditor, in dem der Maskenaufbau als ganz normaler ASCII-Text (also ohne Formatierung und Textattribute) konstruiert wird:

```
Mein Adreßprogramm

Adresse neu anlegen ........................  1
Adresse ändern ..............................  2
Adresse löschen .............................  3
Programm beenden ..........................  0

Ihre Wahl                                  —>
```

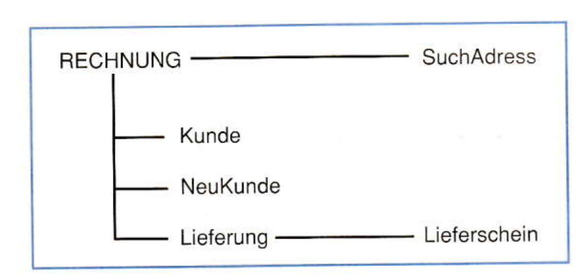
*TALKER* übersetzt dies in folgenden Quellcode:

```
/* ADI-TALKER File:menu.TLK */
/* 24.12.1989 15:45:25 */
@ 0,21 SAY "------------------------------"
@ 1,20 SAY "|       Mein Adreßprogramm      |"
@ 2,21 SAY "------------------------------"
@ 4,23 SAY "Adresse neu anlegen ......... 1"
@ 5,23 SAY "Adresse ändern .............. 2"
@ 6,23 SAY "Adresse löschen ............. 3"
@ 8,23 SAY "Programm beenden ............ 0"
@ 10,23 SAY "Ihre Wahl"
@ 10,46 SAY "->"
```

Nun müssen nur noch oben ein *CLEAR* und unten ein *WAIT TO* eingesetzt werden, und das Menü ist fertig.

**6. Das Hilfsprogramm TCROSS**

*TALK* erlaubt bis zu einer Tiefe von 92 die Verschachtelung der Aufrufe von Modulen (Befehl: "DO ...") aus anderen Programm-Modulen (bzw. Unterprogrammen) heraus. Dabei geht die Übersicht leider sehr schnell verloren. Deshalb gibt es *TCROSS* (Crossreferenz), welches einen Strukturbaum aus dem *TALK*-Programm erzeugt. Beispiel:




**7. Das Hilfsprogramm TMAKE**

Sobald *TALK*-Unterprogramme geschrieben sind, nimmt *TMAKE* deren Titel in eine Textdatei auf. Diese ist quasi eine Liste aller noch uncompilierten Programmfragmente (Module). Jedes Modul hat immer zwei Dateien: Eine Textdatei *.TLK in welcher der lesbare Quellcode (Source) steht und eine Programmdatei *.TLP, die dann den compilierten Objektcode enthält. *TMAKE* prüft nach seinem Aufruf alle Dateien *.TLK und *.TLP und vergleicht deren Erstellungsdatum bzw. -uhrzeit. Ist die Programmdatei älter, muß eine Quelltextänderung vorliegen. Jetzt wird das Modul in *TMAKE* aufgenommen und mit einem "DO"-Befehl eingebunden.

**8. Das Hilfsprogramm TEXPO**

Es kommt nicht so selten vor, daß zwischen mehreren Dateien ein reger Datenaustausch stattfindet. Dieser Datentransport läuft über sogenannte Export- bzw. Importdateien. *TEXPO* erzeugt zu jeder Datenbank automatisch eine Exportdatei, die sämtliche Mermale mit dem Ausgabebefehl >?< auflistet, und eine Importdatei, welche sämtliche Merkmale mit dem "GET"-Befehl verknüpft. Damit läßt sich auch ein regelmäßiges Sichern des Datenbestandes verwirklichen.

**9. Grafikzeichensatz**

Zu dem Hilfsprogramm *TCROSS* existiert eine Konfigurationsdatei *TCROSS.CFG*, in welcher der aktuelle Zeichensatz festgehalten ist. Durch Vertauschen des TOS-Zeichensatzes mit den vielen (leider unnötigen) hebräischen Zeichen gegen den bekannteren IBM-Zeichensatz in dieser Datei, können die Strukturbäume besser anschaulich gemacht werden (durch Rähmchen und Boxen).

**10. Das integrierte Übungshandbuch**

Das vorbildliche Handbuch zu AdiTALK ST plus zeichnet sich (gegen seine Vorgänger) durch bessere Übersichtlichkeit aus. Nicht nur, daß ein recht nützliches Register mitgeliefert wird, es ist auch ein größerer "Einsteigerteil" in Form eines Übungskapitels integriert. Die dortigen Beispiele sind bewußt einfach gehalten, um dem Datenbankneuling die Furcht zu nehmen. Die Beispiele sind so nützlich, daß der Profi durchaus auch dort ein Auge riskieren darf: Gestalten der Bildschirmausgabe, Programme zu neuen zusammenbauen UND (!) typische Programmierfehler erkennen und vermeiden. Ein Referenzteil listet alle Programmbefehle ausführlich mit Schreibweise (Syntax), Wirkung, Erklärung (Semantik), Zusatzschalter und Beispielen auf.

**Datentypunterschiede zwischen INIT und TALK**

| Datentyp | zulässige Größe in TALK | zulässige Größe in INIT |
|---|---|---|
| Typ "C" Zeichenkette | 79 Zeichen | 79 Zeichen |
| Typ "N" Ganze Zahl | Eingabe = 10 Stellen (von -21.474.836,74 bis +21.474.836,74) Ausgabe = 10 Stellen | Eingabe = 12 Stellen Ausgabe = 12 Stellen |
| Typ "D" Dezimalzahl | Eingabe = 9 Vor- und 9 Nachkommastellen Ausgabe = 10 Vor- und 9 Nachkommastellen | Eingabe = 7 Vor- und 7 Nachkommastellen Ausgabe = 7 Vor- und 7 Nachkommastellen |
| Typ "S" Geldbetrag | Eingabe = 7 Vor- und 2 Nachkommastellen Ausgabe = 8 Vor- und 2 Nachkommastellen | Eingabe = 9 Vor- und 2 Nachkommastellen Ausgabe = 9 Vor- und 2 Nachkommastellen |

*Zwischen der Deklaration in INIT und der Ausführung durch TALK sind Unterschiede in der Dimensionierung von Datenfeldern zu beachten.*

## Wichtig für TALK(Alt)-Besitzer

Ein Upgrade von Version 2.XX kostet 150 DM (inkl. MWSt.) und erfordert die Einsendung der Originaldisketten und der Registrierkarte. Für lediglich 100 DM kommen die Besitzer der AdiTALK-Versionen von vor dem 31.12.1988 (Version 2.1) in den Genuß einer Treueprämie. Gleiches gilt, wenn ein Upgrade für Adimens ST plus bereits in Anspruch genommen wurde. Wenn sich mindestens 3 Anwender (mit 3 Originalversionen!) zu einer Upgrade-Aktion (mit einer Liefer- und Rechnungsadresse) zusammenschließen, belohnt ADI diese Zusammenarbeit mit einem "Anwenderbonus" und, es kostet jeden Anwender auch nur 100 DM, weil die Auslieferung dadurch sehr vereinfacht wird.

## Fazit

AdiTALK ist in wesentlichen Punkten entscheidend verbessert worden. Ganz offensichtlich hat man sich bei ADI Software wichtige Anregungen aus der Anwenderschaft zu Herzen genommen und sie in dieser Version verwirklicht. AdiTALK soll den alten Adimens-Nutzern keine Konkurrenz mit ausgefeilten Möglichkeiten bieten, vielmehr wird der Anwendungsschreiber viele interessante Tools für seine Arbeit finden.

*DK*

*Bezugsadresse:*

*ADI Software GmbH*
*Hardeckstraße 5*
*7500 Karlsruhe*
*Telefon: 0721/570000*

# GFA-BASIC 3.5

## Matrizenoperationen inklusive

**Wer bisher versuchte, Matrizen mit GFA-BASIC 3.0 auszurechnen, mußte auf selbstgeschriebene Programme zurückgreifen. Ab sofort gehört die umständliche Rechnerei der Vergangenheit an: GFA Systemtechnik liefert nun die Version 3.5 ihrer Programmiersprache GFA-BASIC aus, bei der Matrizenbefehle bereits zum festen Wortschatz gehören.**

Die Beweggründe von GFA Systemtechnik, eine Programmiersprache mit solch speziellen Befehlen herauszugeben, liegen im Dunkeln. Jeder Schüler, der einmal mit Matrizen gearbeitet hat, wird GFA-BASIC 3.5 allerdings nie mehr missen mögen, denn es ist eine recht große Erleichterung, den Computer die Rechenaufgaben lösen zu lassen. Wozu werden Matrizen überhaupt gebraucht? In kurze Worte gefaßt, kann man sagen, daß man sie bei der Berechnung komplexer Zahlen braucht und lineare Gleichungssysteme mit ihnen lösen kann. Sehen wir uns doch mal ein Gleichungssystem an:

$$\begin{aligned} 3x + 2y + 4z &= 6 \\ 2x - 4y + z &= 2 \\ x + 3y - 4z &= 3 \end{aligned}$$

Ein solches Gleichungssystem wird gewöhnlich durch eine Matrix, die sogenannte Koeffizientenmatrix, die den Koeffizienten vor den Unbekannten enthält, und einen Spaltenvektor, der die linearen Anteile der Gleichung enthält, dargestellt. Das sieht dann so aus:

$$\begin{vmatrix} 3 & 2 & 4 \\ 2 & -4 & 1 \\ 1 & 3 & -4 \end{vmatrix} \quad \begin{vmatrix} 6 \\ 2 \\ 3 \end{vmatrix}$$

Multipliziert man nun die Koeffizientenmatrix mit einem Vektor, der die Unbekannten enthält, ergibt sich:

$$\begin{vmatrix} 3 & 2 & 4 \\ 2 & -4 & 1 \\ 1 & 3 & -4 \end{vmatrix} * \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} 3x + 2y + 4z \\ 2x - 4y + z \\ x + 3y - 4z \end{vmatrix}$$

Die rechte Seite der Gleichung kennen wir bereits, wir können sie ersetzen:

$$\begin{vmatrix} 3 & 2 & 4 \\ 2 & -4 & 1 \\ 1 & 3 & -4 \end{vmatrix} * \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} 6 \\ 2 \\ 3 \end{vmatrix}$$

Um den Lösungsvektor zu isolieren, multiplizieren wir mit der Inversen der Koeffizientenmatrix und es ergibt sich:

$$\begin{vmatrix} x \\ y \\ z \end{vmatrix} = \text{INV} \begin{vmatrix} 3 & 2 & 4 \\ 2 & -4 & 1 \\ 1 & 3 & -4 \end{vmatrix} * \begin{vmatrix} 6 \\ 2 \\ 3 \end{vmatrix}$$

Nun stehen auf der rechten Seite nur noch bekannte Größen und man kann die Matrix ausrechnen, oder man läßt es GFA-BASIC tun. Soviel also zur grundsätzlichen Behandlung von Matrizen. Inwiefern kann man nun die Matrizen mit der neuen Version 3.5 berechnen und manipulieren? Dazu stehen eine Menge neuer Befehle zur Verfügung, die wir jetzt betrachten werden.

### Matrizenbefehle

Alle neuen Matrizenbefehle beginnen mit der Abkürzung *MAT*, was es recht einfach macht, sich den neuen Sprachschatz schnell einzuprägen. *MAT ABS* erzeugt bei allen Komponenten einer Matrix positive Zahlen, indem die Vorzeichen gelöscht werden. Natürlich lassen sich Matrizen bzw. Vektoren auch addieren. Dabei können entweder Zahlen oder andere Matrizen auf eine Matrix addiert werden, auch die umgekehrte Form, also die Subtraktion, ist durch einen einzigen Befehl leicht zu bewerkstelligen. Auch andere Rechenoperationen wie Multiplizieren und Negieren. Weiterhin lassen sich die Matrizen kopieren und löschen.

Etwas mathematisch muß ich leider doch werden, um Ihnen alles erklären zu können. Wie sollte ich sonst begreiflich machen, daß man mit dem Befehl *MAT DET* die Determinante einer quadratischen Matrix ausrechnen kann? Dabei ist die Determinante gleich Null, wenn sich die Matrix invertieren läßt. Vorteilhaft: Eine Matrix läßt sich mit einem einzigen Befehl auf Diskette bzw. Platte abspeichern und wieder einlesen. Dazu existieren die Befehle *MAT INPUT #* und *MAT PRINT #*.

Matrizen kann man auch normieren. Normieren heißt, daß die Summe der Quadrate aller Komponenten in der jeweiligen Zeile/Spalte gebildet wird. Dann werden alle Komponenten durch die Wurzel der Summe geteilt. Nach der Normierung ist die Summe der Quadrate der jeweiligen Komponenten der Zeile/Spalte Eins.

Bevor ich nun alle Befehle eingehend erläutere, sei den Mathematik-Freaks, Studenten und Gymnasiasten, die sich mit Matrizen beschäftigen (müssen), empfohlen, sich den Befehlsumfang der neuen GFA-BASIC-Version einmal bei einem Fachhändler anzuschauen.

### Weitere neue Befehle

Nicht nur für "Matriker" stellt die neue Version die entsprechenden Befehle zur Verfügung, auch vier andere mathematische Befehle haben Platz in der 3.5-Version gefunden. Ab sofort kann beispielsweise die Fakultät (n!) einer Zahl berechnet werden. Dazu existiert jetzt der Befehl *FACT(n)*. *y=VARIAT(n,k)* wiederum berechnet die Anzahl der Variationen von n Elementen zur k-ten Klasse ohne Wiederholung und schreibt diesen Wert in die Variable y. Der letzte neue Rechenbefehl

nennt sich *COMBIN*. Dieser Befehl berechnet den Binominalkoeffizienten, also die Anzahl von Kombinationen von n Elementen zur k-ten Klasse ohne Wiederholung.

Als letzte Neuerung bei den Befehlen sei *_DATA* erwähnt: Er gibt an, wo der DATA-Zeiger steht. Der Datenzeiger kann allerdings auch gesetzt werden, indem *_DATA* ein Wert zugewiesen wird.

## Neuerungen im Editor

Das Scrolling im Editor ist wesentlich schneller geworden. Das wird dadurch erreicht, daß für jede Programmzeile zwei Bytes mehr benötigt werden, in der einige Informationen für den Interpreter stehen. Lange Programme brauchen dadurch natürlich auch wesentlich mehr Speicherplatz im Editor. Normalerweise wird der Programm-Code aber nicht so lang, daß man dadurch einen Nachteil erleidet. Ein mit der 3.5-Version compiliertes Programm ist dagegen genau so lang wie ein mit der 3.0x-Version compiliertes.

Endlich!! kann auch in den Kopfzeilen geschlossener Prozeduren bzw. Funktionen gesucht werden, d.h. bei der Suchfunktion mit *F6* bzw. *Control-F* wird auch dann etwas gefunden, wenn der gefundene Begriff in der ersten Zeile einer eingeklappten Prozedur steht. Sie lesen richtig, auch Funktionen können jetzt durch Druck auf *HELP* eingeklappt werden.

Beim Auflisten werden Labels, also Sprungmarken, um zwei Zeichen nach links eingerückt dargestellt (wie bei *SELECT CASE*). Die Logik dieser Funktion wage ich zu bezweifeln. Labels, die sowieso schon am linken Rand stehen, können beispielsweise nicht mehr eingerückt werden, da sie ja dann "aus dem Bildschirm fallen" würden.

Mit *Tab*, *Control-Tab*, *Shift-links-Tab* und *Shift-rechts-Tab* kann ab sofort zur nächsten oder vorhergehenden Tabulatorposition gesprungen bzw. Leerzeichen bis zur nächsten oder vorhergehenden Tabulatorposition eingefügt werden.

Natürlich wird auch ein 28seitiges Handbuch-Update zum Einheften in den 3.0x-Ordner mitgeliefert, in dem alle neuen Befehle eingehend beschrieben sind.

## Preise

Für all diese Neuerungen müssen DM 70,- als Upgrade-Preis bezahlt werden. Zum Upgrade von Version 3.0x auf Version 3.5 sind beide Originaldisketten (Compiler- und Interpreter-Diskette) an GFA Systemtechnik einzuschicken. Weitere Auskünfte hierzu erhalten Sie bei GFA Systemtechnik in Düsseldorf. Wer noch kein GFA-BASIC besitzt, kann das Entwicklungssystem 3.5 für DM 268,- inklusive Compiler kaufen.

Für alle Mathematiker, Schüler und Studenten ist die Version 3.5 sicher eine lohnende Anschaffung. Zum einen ist der Editor wesentlich schneller im Scrolling geworden, was sich besonders bei längeren Programmen positiv bemerkbar macht. Zum anderen sind die neuen Befehle doch recht komfortabel und für jeden, der sich mit Matrizen beschäftigt, unentbehrlich.

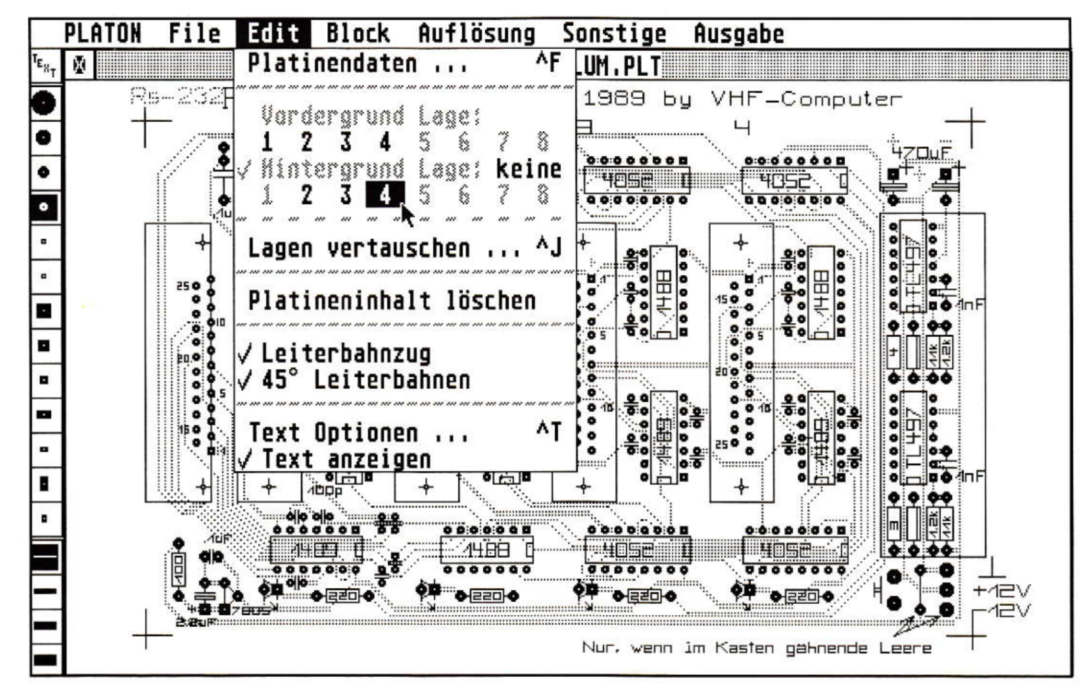
*MP*

*Bezugsadresse:*

*GFA Systemtechnik GmbH*
*Heerdter Sandberg 30-32*
*4000 Düsseldorf 11*
*Tel. (0211) 5504-0*

# PLATON

## Leiterplatten schnell entworfen




**Das Leiterplatten-CAD-System Platon von VHF-Computer stand uns in seiner Version V1.4 zur Verfügung. Es ist insbesondere auf die Belange des gehobenen Hobby- und des semiprofessionellen Bereichs zugeschnitten. Zum Preis von DM 298.- sind im Lieferumfang zwei Disketten und ein Handbuch in Form eines DIN A5-Ordners enthalten. Das Programm ist nicht kopiergeschützt, was für Festplattenbenutzer interessant ist. Nebenbei kann man so Sicherheitskopien zum eigenen Gebrauch leichter anfertigen.**

Auf der Diskette sind neben den Haupt- und Installationsprogrammen noch weitere Dateien vorhanden: Zeichensätze, Drucker- und Plotter-Treiber, Bauteilebibliotheken, einige Beispielplatinen sowie Utilities. Unter diesen befindet sich das hervorragend gemachte Profi-Dateiauswahlfenster als Accessory, gegen das selbst TOS 1.4 spartanisch wirkt.

Besitzer der alten Platon-Version V1.2 können ihre Platinen mit einem Konvertierungsprogramm in das neue Format übernehmen. Es können auch Platinen von "PCB-Layout" und "Platine-ST" ins Platon-Format umgewandelt werden.

Der sehnlichste Wunsch vieler (Hobby-) Elektroniker, ein sehr guter Autorouter zu einem günstigen Preis, wird hier (leider) nicht erfüllt. Ein 'Low-Cost'-Autorouter wäre wohl mehr Last als Hilfe, da er die einfachsten Forderungen nach effizientem, hochfrequenztauglichem Layoutdesign gar nicht oder nur unzureichend erfüllen könnte. Aus diesem Grunde arbeitet der Editor des Programms manuell.

Platon V1.4 ist ein vektororientiertes Programm zur computergestützten Erstellung und Ausgabe von Leiterplatten-Layouts und ist in Verbindung mit einer XYZ-Fräsbohranlage zum Bohren von Platinen vorgesehen. Selbstverständlich ist das Programm vollständig mausgesteuert und GEM-unterstützt. Dies erlaubt kurze Einarbeitungszeiten auch für ungeübte Benutzer.

Das Programm kann Leiterplatten mit einer Auflösung bis zu 1/320 Zoll edieren und intern mit 1/2000 Zoll verarbeiten. Mit diesen Leistungsdaten ist es möglich, z.B. die Stekkerleiste des ROM-Ports der ATARI-Computer zu zeichnen, was nicht selbstverständlich ist, da dieser nicht genormte Maße hat.

Das Handbuch der Version 1.3 umfaßt mehr als 100 Seiten und führt Sie auf leichtverständlichem Weg in die ersten Schritte zur Erstellung einer Platine ein. Anhand eines im Handbuch abgedruckten Schaltplans wurde Schritt für Schritt das zugehörige fertige Layout erzeugt. Auf diese Weise kann sich der Anwender am besten in das Programm einarbeiten. Für die getestete Version 1.4 war das Handbuch leider noch im Druck und konnte deswegen nicht in Augenschein genommen werden. Mit Erscheinen dieses Artikels sollte es aber fertiggestellt sein.

Im Hauptteil der Dokumentation werden die einzelnen Menüpunkte mit Abbildungen ausführlich erklärt, denn ein Bild sagt mehr als tausend Worte.

```
PLATON
  PLATON
  Harddiskparken
  Dateiauswahl

File
  neue Platine ...              ^N
  Platine laden ...             ^L
  Platine speichern             ^S
  Platine speichern als ...     ^H
  Einstellung laden ...
  Einstellung speichern ...
  Programm beenden              ^Q

Edit
  Platinendaten ...             ^F
  Vordergrund Lage:
  1 2 3 4 5 6 7 8
  Hintergrund Lage: keine
  1 2 3 4 5 6 7 8
  Lagen vertauschen ...         ^J
  Platineninhalt löschen
  ✓ Leiterbahnzug
  ✓ 45° Leiterbahnen
  Text Optionen ...             ^T
  ✓ Text anzeigen

Block
  Block ausschneiden            ^X
  Block kopieren                ^C
  Block einfügen                ^V
  1 2 3 4 5 6 7 8
  Block löschen                 ^E
  Block ändern ...              ^A
  Block speichern ...           ^W
  Block laden ...               ^R
  Block spiegeln                ^M
  Block Optionen ...            ^Z
  Blockverwaltung ...           ^K

Auflösung
  Auflösung:
    1/20  Zoll                  ^1
    1/40  Zoll                  ^2
  ✓ 1/80  Zoll                  ^3
    1/160 Zoll                  ^4
    1/320 Zoll                  ^5
    1/1   mm                    ^6
    1/10  mm                    ^7
  Cursorschrittweite:
    1/10 Zoll
  ✓ 1/20 Zoll
    1/40 Zoll
    1/80 Zoll

Sonstige
  Statistik ...                 ^I
  Einstellungen ...             ^G
  Anzeige ...                   ^Y

Ausgabe
  Weg optimieren...             ^O
  Plotten ...                   ^P
  Drucken ...                   ^D
  Gerber-Ausgabe ...
  Bohr-Daten ...
  Bohren, Fräsen ...            ^B
  Benutzerprogramm ...          ^U
```

*Die Menüleiste Platons*

Auch der Disketteninhalt und das umfangreiche Stichwortregister sind in dem ansehnlichen Plastikordner zu finden, sie 'entschärfen' doch die Situationen, bei denen man nicht mehr weiß, wie es weitergeht. Zu guter Letzt sind die möglichen Fehlermeldungen und die zugehörigen Maßnahmen aufgeführt.

## Installation

Zum Betrieb sind ein monochromer Monitor und ein doppelseitiges Diskettenlaufwerk notwendig. Wenn es sein muß, läuft das System auch mit RAM-TOS, was natürlich nicht empfehlenswert ist. Besitzer eines Laserdruckers benötigen mindestens das Blitter-TOS. Es versteht sich fast von selbst, daß man für so ein Programm mindestens ein MByte Speicher haben sollte. Das CAD-System läuft zwar mit 512 kByte RAM, doch einige wichtige Funktionen werden noch vor ihrer Ausführung unterbrochen. So ist es z.B. nicht möglich, mit einem Drucker reprofähige Ausdrucke zu erhalten, da die Berechnung dieser Ausgabe immensen Speicherplatz benötigt.

Leider unterstützt das Programm nicht den Farbmodus, in dem man mehrere Lagen einer Platine schön übersichtlich darstellen könnte. Dies scheitert aber wohl an der zu niedrigen Farbauflösung der ST-Modelle. Mit einer hochauflösenden Farbgrafikkarte könnte man dem vielleicht abhelfen, zum Test stand mir leider keine zur Verfügung. Laut Handbuch ist das Programm in der Lage, mit einem Großmonitor zu arbeiten.

Beim Austesten dieses Programmes mit dem neuen Rainbow-TOS zeigt es jedoch leider einige Fehler: Die gezeichneten Bauteile und Leiterbahnen lassen sich nicht mehr mit Doppelklick der Maustaste löschen. Auch beim Aufruf der Accessories verhalten sie sich nicht immer wie gewünscht. Es ist also anzunehmen, daß sich der Programmierer nicht ganz an die von ATARI vorgegebenen Konventionen bei der GEM-Programmierung gehalten hat.

Beim Installieren von Platon V1.4 können fast alle Möglichkeiten der Plotter, Drucker und XYZ-Anlagen eingestellt werden, so daß auch exotische Ausgabegeräte angeschlossen werden können. Leider sind nur die Druckertreiber für die Epson-Drucker FX80 und LQ1000 sowie HP Laser-JetII mitgeliefert. Beim Laser-Jet-Treiber ist es schade, daß er nicht mit dem HP-Desk-Jet funktioniert. Der Treiber geht von einem Seitenspeicher aus, der dem Desk-Jet leider nicht zur Verfügung steht. So kommt es, daß die Leiterbahnen auf zwei Druckzeilen komprimiert werden, die Lötaugen dann jedoch richtig auf dem Papier erscheinen. Vielleicht läßt sich da ja noch etwas machen. Besitzer anderer Drucker, vor allem von NEC und ATARI-Laser SLM804, müssen die Druckertreiber selber vor der Installation konfigurieren. Diese Arbeit sollte bei jedem guten Programm überflüssig sein. Mit der Ansteuerung der Laserdrucker sind perfekte und schnelle Ausdrucke möglich. Der Plotter-Treiber unterstützt u.a. einen HP 7475-Plotter.

## Der Editor

Nach dem Starten des Programms kommt man sofort in den Arbeitsbildschirm. Die Platinen können in folgenden Auflösungen dargestellt werden: 1/20, 1/40, 1/80, 1/160, 1/320 Zoll sowie 1/1 und 1/10 mm. Die Millimeterauflösungen sind bestimmt sinnvoll, um u.a. die Gestaltung von Frontplatten zu erleichtern. Wenn man von einer Auflösung in eine andere umschaltet, wird die letzte Mausposition als neue Bildschirmmitte definiert. Dies ist bei großen Platinenformaten sinnvoll, um die Übersichtlichkeit zu bewahren, da Platon V1.4 Platinen mit einer Größe von bis zu 832*832 mm verarbeiten kann.

Die Menüleiste des Editors wird nur dann angezeigt, wenn sich der Maus-Cursor auf ihr befindet. D.h. solange Sie sich im Arbeitsfeld befinden, erscheint statt dessen die Infozeile mit den Lagebezeichnungen und Cursor-Positionen. Letztere können absolut oder relativ zum wählbaren Nullpunkt angezeigt werden. Sieben Menütitel und eine Symbolleiste verbergen die Funktionsvielfalt des Leiterplatteneditors. Diese Funktionen werden mit der Maus selektiert. Die meisten Menüeinträge können aber auch über die Tastatur mit der 'Control'-Taste plus dem entsprechenden Buchstaben erreicht werden.

Im Menüpunkt "File" hat man die Wahl zwischen Platinen bzw. Einstellungen laden/speichern sowie neue Platine erstellen. Die Einstellungen sind u.a. die Pfadnamen (eine wirkliche Erleichterung für Festplattenbesitzer), die Cursor-Schrittweite, Anzeigenparameter und Blockoperationen etc. Es werden auch Leiterbahn- und Lötpunktsymbole mit abgespeichert, die sich in der Symbolleiste am linken Rand der Arbeitsfläche befinden. Diese Symbole (17 an der Zahl) können nach Ihren persönlichen Bedürfnissen bezüglich der Leiterbahnbreite (0,1 bis 5,0 mm) oder des Durchmessers bzw. der Form der Lötaugen (quadratisch, rechteckig, rund) eingestellt werden.

Beim nächsten Menütitel "Edit" werden die Platine und die Darstellung der Lagen beeinflußt. Einzelne Lagen können gelöscht, Leiterbahnführungen in 45°-Schritten eingestellt werden. Hier können Sie dem Editor ebenfalls mitteilen, welche der bis zu acht möglichen Lagen dargestellt werden soll. Mit der Taste 'Alt' und einer Zahl kann man die Vordergrundlage, und zusammen mit der 'Shift'-Taste die Hintergrundlage auswählen. Das Hintergrundraster läßt sich ein- und ausschalten. Bei der Hintergrundfarbe besteht die Wahl zwischen drei Farben

(Grau, Schwarz oder keine). Im Zeitalter der rasanten Entwicklung im Bereich der Elektronik müßte ein Layout-Programm die SMD-Pads verwalten können. Bei diesem Programm können Sie auch die Lagen für die SMD-Bestükkung angeben, und zwar unabhängig von der oben erwähnten Bestückungslage. Die winzigen Lötflächen werden dann genauso behandelt wie die herkömmlichen Lötaugen, mit dem Unterschied, daß hier keine Bohrungen vorgesehen sind (logo).

## Blockoperationen

Das Programm bietet im Menütitel "Block" folgende Funktionen: Ausschneiden, Kopieren, Einfügen, Löschen, Ändern, Speichern, Laden, Spiegeln sowie Optionen und Blockverwaltung. Sollen die Blockoperationen nur auf die Leiterbahnen und Texte wirken, die sich vollständig im eingerahmten Block befinden, so teilen Sie dies dem Editor im Punkt "Optionen" mit, andernfalls haben diese Operationen auch Einfluß auf die Elemente, von denen nur ein Teil im Block liegt. Auch eine Einschränkung der Aktionen auf einzelne Lagen oder Kriterien ist möglich, um z.B. eine spezielle Lage, die nur Texte enthält, vor ungewollten Veränderungen zu schützen. Mit den Kriterien 'Lötpunkte', 'Text' und 'Leiterbahnen' wirken die Blockoperationen dann nur noch auf die ausgewählten Elemente.

Eine Besonderheit weist die Funktion "Block verschieben" auf: Auf Wunsch können überstehende Leiterbahnen wie Gummibänder nachgezogen werden. Die Endpositionen von Leiterbahnen, die außerhalb des Blockbereichs liegen, bleiben beim Verschieben an ihrer ursprünglichen Position. Mit der Blockverwaltung von Platon können Sie auf fast jede Taste der Tastatur einen Block legen. Diese Blöcke werden nach Drücken der entsprechenden Taste automatisch geladen und auf den Bildschirm gebracht. Auf diese Weise können Sie häufig verwendete ICs schnell ins Layout einfügen. Die gesamten Tastenbelegungen für die Blöcke lassen sich auf Festplatte speichern und wieder einladen. So können Sie für verschiedene Situationen eine eigene Belegung definieren: eine für SMD, die nächste für Frontplattenelemente usw.

*Platons Druckoptionen*

## Texte

Einer der aufwendigen Programmteile ist sicherlich der Texteditor. Er bietet äußerst komfortable und vielfältige Möglichkeiten, Texte auf Platinen, Frontplatten oder ähnlichem zu gestalten. Um diese auch vollständig ausschöpfen zu können, gestattet Ihnen Platon V1.4 neben den mitgelieferten Zeichensätzen auch noch eigene Kreationen mittels beigelegtem Zeichensatzeditor. Sie können insgesamt vier verschiedene Zeichensätze gleichzeitig laden und verwenden, was für die Praxis vollkommen ausreicht.

Mit der Maus stellen Sie die Höhe, Breite, Neigung, Dicke und den Abstand der Zeichen ein. Sobald Sie irgendwelche Einstellungen ändern, sehen Sie die Auswirkungen sofort. Die Texte können auch gespiegelt und in acht verschiedenen Richtungen geschrieben werden. Für jeden Zeichensatz können Sie die oben eingestellten Werte extra speichern. Da Platon V1.4 in der Hochsprache C geschrieben wurde, haben diese vielfältigen Funktionen einen Nachteil: Mit jedem Buchstaben mehr auf der Platine wird der Bildschirmaufbau langsamer. Um dennoch wieder flüssig arbeiten zu können, haben Sie die Möglichkeit, die Textausgabe auf dem Bildschirm zu unterdrükken. Bevor man eine Platine erstellt, gibt man ihre Größe sowie die Anzahl der Lagen ein . Man kann hier angeben, welche der Lagen einer Bestückungs- oder Lötseite zugeordnet sind. Häufig ist es hilfreich, die Abmessungen der Platine etwas großzügiger anzugeben. Dadurch können Sie auch außerhalb der Leiterplattenbegrenzung Beschriftungen oder überstehende Bauteile anbringen. Auch nach dem Entwurf einer Platine läßt sich das Fenster jederzeit auf das tatsächliche Format verkleinern bzw. vergrößern.

Die Bestückungspläne werden mit Hilfe der Bauteilebibliotheken erstellt. Die Elemente in diesen Bibliotheken lassen sich auch vom Benutzer selbst definieren bzw. hinzufügen. Sie können seltene, exotische Bauteile mit 'Leiterbahnzügen' zeichnen, ohne daß diese mit in die Bauteileliste abgespeichert werden müssen.

Das Programm bietet leider (noch) keine Möglichkeit, Bestückungslisten auszugeben. Doch für den ehrgeizigen Anwender hat der Programmierer ein sogenanntes 'Hintertürchen' eingebaut, das im Menüpunkt 'Benutzerprogramme' zu finden ist. Damit können Sie Ihr heißgeliebtes Textverarbeitungsprogramm oder schlicht ein abwechslungsreiches Spiel einladen und später dann wieder zum Editor zurückkehren ...

Apropos Benutzerprogramme: Im Anhang ist eine Erklärung des Dateiformats einer '*.PLT'-Datei abgedruckt, mit deren Hilfe man das Layout in eigene Programme einbinden kann, um daraus z.B eine Schaltungssimulation durchführen zu lassen usw.

## Ausgabe

Die Platinen-Layouts können auch in Form von Lötstoppmasken oder nur als einzelne Lagen auf einem Zeichengerät (Drucker, Plotter) ausgegeben werden. Auf dem Drucker ist die Ausgabe in Originalgröße bis hin zur vierfachen Vergrößerung vorgesehen. Beim Plotter ist der Maßstab sogar stufenlos von 0,1:1 bis 10:1 einstellbar. Mit dem Druck- bzw. Plot-Programm können Sie wählen, ob Texte, Leiterbahnen, Lötpunkte und/oder Bohrerzentrierungen zum Gerät gesendet werden. Beim Plotter besteht auch die Möglichkeit, die Geschwindigkeit der Stiftbewegungen sowie die Nummer des Stiftes bzw. die Stiftdicke festzulegen. Um z.B. mit Tuschestiften auf Folie sehr gute Ergebnisse zu erzielen, sollte wesentlich langsamer geplottet werden können als auf Papier. Die besten Ergebnisse werden - laut Handbuch - mit einer mattierten Zeichenfolie und tiefschwarzer Tusche (z.B. Staedtler 747 TL-9) erreicht. Beim Drucker muß eine Entscheidung zwischen einem schnellen Probeausdruck und dem reprofähigen Qualitätsdruck getroffen werden. Dies gilt auch für 9-Nadeldrucker, deren Ausdrucke von beachtlich hoher Schärfe sind. Leider muß

man auf solche Ausdrucke ziemlich lange warten, da der Druckkopf viermal über eine Zeile fährt, um einerseits eine hohe Punktdichte und andererseits eine ausreichende Lichtundurchlässigkeit zu erreichen.

## Gerber-Fotoplotter

Als eines der herausragenden Merkmale eines Leiterplatten-Editors im ST-Sektor besitzt Platon V1.4 einen Treiber zur Erzeugung einer Datei im Gerberformat. Sie können mit diesem die Daten Ihrer Platine auf Diskette speichern und von einem entsprechendem Dienstleistungsunternehmen eine Folie mit extrem hoher Kantenschärfe herstellen lassen. Durch dieses Verfahren werden mit Abstand die besten Ergebnisse erzielt. Da Codes in der Gerberdatei nicht festgelegt sind, wie z.B. beim HP-GL für Plotter, lassen sich diese in der Aperture-Tabelle definieren. Es stehen insgesamt 90 D-Codes zur Verfügung. In dieser Tabelle wird festgelegt, welche Blendenformen und -öffnungen der Fotoplotter für den Plott verwendet werden. Mit dem mitgelieferten Vektorkonvertierungsprogramm können fremde Gerber-Daten eingelesen und dann dieses Layout mit dem Editor weiterverarbeitet werden.

## Fräsbohranlage und Bohrdaten

Bei der Ansteuerung einer Fräsbohranlage, die werksseitig auf eine "isel 3-Achsen-Schrittmotorsteuerung" installiert wurde, können auch alle nur erdenkbaren Möglichkeiten ausgenutzt werden. Das sind z.B. die Fräsernummern, -durchmesser, Eintauchtiefe, Senk- und Frägsgeschwindigkeit sowie Tiefenanschlag beim Bohren, um nur einige zu nennen. Auch eine Referenz- oder Nullfahrt ist möglich. Erstere fährt zur Nullposition des Fräsers, während letztere den Fräser zur einstellbaren Startposition des Programmes führt.

Es sind zehn verschiedene Startpositionen möglich, die alle per Maus angeklickt werden können. Die Ausgabe (auch auf dem Plotter) ist sogar optimierbar. Bei diesem Vorgang werden beispielsweise die zusammenhängenden Leiterbahnen auch wirklich zusammenhängend bearbeitet oder geplottet. So kann man schnell, komfortabel und vor allem preiswert Prototypen und Kleinserien von Platinen bzw. Frontplatten fräsen und bohren.

Mit einem Bohrdatenprogramm läßt sich eine Datei mit den Bohrdaten erzeugen. Sie können auch, sofern Sie über einen Lochstreifenstanzer verfügen, die Bohrdaten direkt auf Lochstreifen stanzen. Mit diesem Lochstreifen können Sie Ihre Platinen von CNC-Bohrautomaten bohren lassen. Die XYZ-Anlage ist also für den Heimbereich gedacht, während CNC-Automaten mehr zur Herstellung von Serien geeignet sind.

## Bedienung

Ediert wird grundsätzlich mit der Maus. Die Bedienung mit diesem sonst so vertrauten Nager erfordert doch ein wenig Eingewöhnung. Neben der üblichen Menüansteuerung können die Lötpunkte mit der linken Maustaste gesetzt werden. Zum Löschen derselben genügt ein Doppelklick mit der gleichen bzw. mit der 'Alternate'-Taste und einem Einfachklick. Um Linien zu zeichnen und zu löschen, kommt hier ungewohnterweise die rechte Maustaste zur Verwendung. Dadurch vermeidet man lästige Menüaufrufe. Sehr nützlich ist die Möglichkeit, die Lage der Leiterbahnenden nachträglich mit der rechten Maustaste zu verändern (Drag-While-Move). Sie klicken einfach einen Eckpunkt an, und die Leiterbahnen folgen der Maus wie ein Gummiband. Mit der 'Shift'-Taste kann man (in Verbindung mit den jeweiligen Mausoperationen) Blöcke, Texte usw. kopieren. So lassen sich sehr leicht bereits verlegte Leiterbahnen verschieben. Mit der Undo-Funktion können Sie sämtliche Edierfunktionen wieder rückgängig machen, um z.B. falsch gesetzte Lötaugen wieder zu entfernen. Einen Fehler konnte ich beim Austesten des Programmes doch noch entdecken: Wenn man einen Block verschiebt, darf man nicht gleichzeitig die 'Undo'-Taste drücken. Sonst zeichnet der ST nach Beendigung des Verschiebens zwei Bomben. Seltsamerweise befindet man sich danach wieder im Desktop, als ob nichts passiert wäre.

Als recht nützliche Funktion erweist sich die Möglichkeit, beim Speichern die alte Datei in die Extension '*.BAK' umzuwandeln oder einfach zu überschreiben. Des weiteren kann man das Programm so einstellen, daß es bei jedem Neustart mit der zuletzt bearbeiteten Platine und eingestellten Werten erscheint.

Alles in allem ist das Programm sehr umfangreich, doch sollte der Programmierer die Disharmonie mit dem Rainbow-TOS abbauen. Wenn man sich erst einmal an die ungewohnte Maussteuerung gewöhnt hat, kann man mit dem Programm (fast) professionell arbeiten, und es ist in jedem Fall sein Geld wert.

Von VHF-Computer war zu erfahren, daß für die Zukunft u.a. folgende Erweiterungen geplant sind: Umfräsen von Leiterbahnen und Lötpunkten für Prototypenfertigung, Einlesen von HP-GL-Daten, interaktiver Autorouter und Ausfüllen von Masseflächen.

*Martin Macher*

***Bezugsadresse:***

*VHF-Computer*
*Maurener Weg 115a*
*7030 Böblingen*
*Tel.: 07031/272271*



# S.& P.-CHART

## Der feine Analytiker

**Wenn man eine Vielzahl ähnlicher Programme betrachtet, die zudem ein enges Spezialgebiet abdecken, wird man sehr sensibel für Kleinigkeiten. So erging es mir bei der Arbeit mit S.&P.-CHART. Ein Programm zur Chart-Analyse, einfach zu bedienen, unproblematisch, mit Liebe zum Detail. Aber der Reihe nach:**

Erst muß der Kopierschutzstecker in den ROM-Port geschoben werden. (Über Sinn und Unsinn dieser "Antikopiertechnik" soll hier nicht weiter philosophiert werden.) Danach gestalten sich Installation und Programmstart recht unkompliziert. Nach der Abfrage von Datum und Uhrzeit (was nach jedem neuen Programmstart geschieht) erscheint das Eröffnungsbild mit der Desktop-Menüleiste. Nun ist eine Datei mit den Kursdaten zu laden. Diese sind in drei Gruppen aufgeteilt: Aktien, Optionsscheine und "Diverses" (darunter verbergen sich z.B. Indizes, Währungen oder Edelmetalle).

Weniger aufregend sind die Menüpunkte *Technik* und *Titel*. Solche Tätigkeiten wie Bild laden und speichern (auch in dem STAD-kompatiblen "PAC"-Format), oder die Einzeleingabe von Kursen, das Anlegen neuer Titel und Ändern von Kennzahlen, sowie die Berücksichtigung von Kapitalmaßnahmen, dazu zählen z.B. Kapitalerhöhung, Splits, Aktienzusammenlegung, sollten Standard in jedem guten Börsenprogramm sein und braucht fast nicht weiter erwähnt zu werden.

Hervorzuheben bleibt die Parameterbox, bei der das Konzept von S.&P.-Chart, nämlich dem Anwender möglichst viele Freiheiten in der Analyse zu bieten, deutlich wird. In diesem Menüpunkt kann man frei wählbare Werte für Tagesdurchschnittslinie, Momentum und Trendbestätigungsindikator und Overbought/Oversold einstellen. So bleiben die Wahlmöglichkeiten nicht an feste Standardwerte wie z.B. 38/100/200-Tagelinien-GD oder 38-zu-100-T.B.I. gebunden. Noch interessanter gestalten sich die Punkte *Charts* und *Analyse*, die wir uns genauer anschauen wollen:

### Chart

Nachdem aus der Kursdatei eine Aktie mit Hilfe eines Pop-Up-Menüs ausgewählt wurde, kann man sofort einen Chart erstellen lassen. Dazu stehen folgende Darstellungsarten zur Verfügung: Tages-, Wochen- (Balken-), Tag/Wochen-Vergleich (absolut oder relativ), mit eingeblendeten Momentum, TBI, vier verschiedene Oszillatoren oder OVB/OVS-Indexlinien. Zusätzlich kann wahlweise eine 38-, 100-, 200- und x-Tage-Durchschnittslinie, sowie eine Gewinnzone eingeblendet werden.

Ein Infoblock, im oberen Bildteil über dem Chart, beliebig ein- und ausblendbar, enthält neben dem Namen des Papiers und des letzten Kursdatums auch viele Hinweise auf Indikatoren und weitere technische Daten (immer bezogen auf das aktuelle Datum):

1. Höchst- und Tiefstkurs, sowie deren prozentualer Abstand zum aktuellen Kurs;
2. die Werte der vier Trendlinien mit einem Tendenzpfeil (Trendlinie steigend oder fallend);
3. den Gewinn pro Aktie und das rechnerische Kurs/Gewinn-Verhältnis, bzw. Bezugspreis, Aktienkurs, Aufgeld, Hebel, immitierte Stückzahl, Fälligkeit und innerer Wert bei Optionsscheinen;
4. die Dividende und deren Rendite;
5. TBI, Momentum und Beginn der Gewinnzone.

### Analyse

Mit Hilfe des Menüpunktes *Kursblatt* gewinnt der Benutzer sofort einen Gesamtüberblick über alle Werte einer Datei. Dies sind alle Angaben aus dem Infoblock jedes einzelnen Papiers (Aktie, Optionsschein oder "Diverses"). Reichhaltig ist die Auswahl an Listen. Dieser Menüpunkt ist wichtig für Anleger, die etwa neu in einer Branche investieren wollen. Ein Beispiel: Jemand hat zuvor aufgrund der Branchenindizes den Chemiebereich favorisiert und möchte nun einen Titel auswählen. Er lädt als erstes die Chemiebranche ein und sucht nun mit der Funktion *Listen* nach Aktien (oder Optionsscheinen), die in den letzten Tagen, Wochen, Monaten oder Jahren (je nach Mentalität, ob Langfrist- oder Kurzfrist-"Moneymaker") noch nicht sehr viel gestiegen sind, oder die 38-, 100-, 200-, bzw. x-Tagelinie durchbrochen haben. Weiterhin wären noch bestes Kurs/Gewinn-Verhältnis, günstigste Dividentenrendite und bestes Momentum als Liste tabellarisch zu erhalten.

Weitere Analysen sind über ein ausgeklügeltes System von Parametern und Bewertungsfaktoren möglich. So kann der Bediener aus einer Liste von 10 Indikatoren, individuell nach seinen Bedürfnissen jene zusammenstellen, die er für maßgebend hält. Zusätzlich versieht er die gewählten Indikatoren mit einer Bewertungszahl, je nach persönlicher Einschätzung für die Gewichtung des Faktors.

Diese Parameter und Bewertungsziffern können separat für jenden Titel abgespeichert werden. Nach dieser Voreinstellung errechnet S.&P.-Chart unter dem Menüpunkt *Analyse* Kauf- und Verkaufssignale für das ganze, zur Verfügung stehende Kursmaterial des gewählen Papieres. Diese sind dann entweder grafisch mit "K" für Kauf und "V" für Verkauf in den Chart eingezeichnet, oder als Liste mit Gewinn/Verlustbilanz ausgegeben.

Ein Beisiel: Der Anleger möchte einen bestimmten Wert genauer im Auge behalten und auch kaufen, wenn sich eine günstige Gelegenheit ergibt. Er wählt im Menü die *Kaufsignale* an und bestimmt die Kriterien in etwa so: 1. Durchbruch der 200-Tage-Linie um 2% sind ihm 3 Punkte wert; 2. Momentum über Null wird mit 2 Punkten gewichtet und 3. 100-Tage-Oszillator über Null soll auch mit 2 Punkten in die Rechnung eingehen.

Die Punktzahl bei der gekauft wird kann noch frei bestimmt werden, d.h. legt man zum Kauf fünf Punkte fest, so muß Kriterium 1 auf jeden Fall, und mindesten eines der anderen auch noch erfüllt sein. Genauso wird mit den Verkaufssignalen verfahren.

Die Auswertung all dieser Kriterien erfolgt dann in dem Menüpunkt *Chart-Ana lyse*. Hier werden jetzt alle zur Verfügung stehenden Kurse bis ca. sechs Jahre zurück nach Signalen durchforstet und Käufe bzw. Verkäufe mit dem entsprechenden Gewinn oder Verlust angezeigt. Man kann nun so lange an den Kriterien feilen, bis weniger oder keine Minuskäufe mehr sichtbar sind. Signale die sich so über Jahre bei einem Papier bewährt haben, geben sicherlich in der Zukunft eine große Hilfe - und um die Zukunft geht es ja schließlich beim Spekulieren.

## Toolbox

Ein Chart soll nicht nur aus Kursverläufen, Durchschnittslinien und nüchternen Zahlen bestehen. Anschaulicher wird eine typische Entwicklung, d.h. die Ausbildung eines Trends oder Formation (siehe auch Grundlagenartikel in Heft 2/90) durch deren grafische Einbindung in den Chart. Um nun diese Formationen (Trendkanäle, "W"-, "M"-, Dreieck-, Rechteckformen) in ein Chartbild einzeichnen zu können, gibt es in S.&P.-CHART die *Toolbox*. Ein Druck auf beide Maustasten ruft die Toolbox an jede beliebige Stelle im Chart auf. Das Zeichnen von Linien (einfach, mehrfach parallel, vertikal und horizontal), Kreise und Vierecke wird hierüber ausgelöst. Auch Texte für Kommentare, Schriftgröße und Drehwinkel, sowie Strichdicke sind dort anwählbar.

Ein Pufferspeicher nimmt Bildteile auf, die dann in anderen Charts frei wieder positioniert werden können. Es kann mit dieserdurch beispielsweise sehr leicht eine Aktie mit einem Aktienindex verglichen werden. Mit der Funktion *Kursbestimmung* placiert man kleine Boxen mit Hinweispfeil, die die Höhe einzelner Kurspunkte im Chart kennzeichnen. Die Zoomfunktion erlaubt das beliebige Vergrößern einzelner Kursabschnitte, die dann automatisch maßstabsgerecht neu entstehen.

**Datei**
- Akt. Laufwerk
- ...eröffnen
- ...einladen
- ...sichern
- ...löschen
- Beenden

**Technik**
- Hardcopy
- Chartbild laden
- Chartbild sichern
- Chartbild löschen
- Freier Speicher
- Uhrzeit
- Parameterbox

**Titel**
- ...neu aufnehmen
- ...löschen
- Kurse anzeigen
- Kurse ändern
- Kurse eingeben
- Tageskurspflege
- Kennzahlen ändern
- Kapitalerhöhung

**Charts**
- Tageschart
- Wochenchart
- Tages- u. Wochenchart
- Chartvergleich
- Momentumchart
- T.B.I.-Chart
- Oszillator
- x-Tage-Linie
- 38-Tage-Linie
- 100-Tage-Linie
- ✓ 200-Tage-Linie
- Gewinnzone
- ✓ Infoblock

**Analyse**
- Kursblatt
- Listen
- Toolbox
- Kaufsignale
- Verkaufsignale
- Chartanalyse
- Param. Laden
- Param. Speichern

*Bild 1: Die Menüs von S.&P.-CHART*

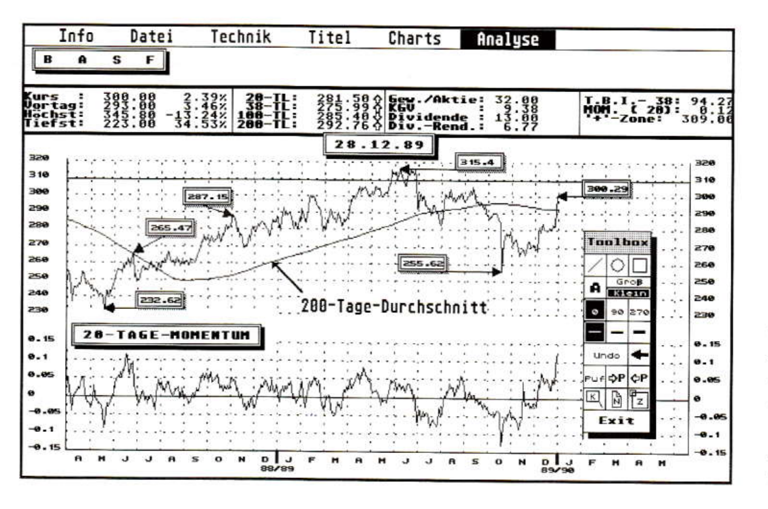



*Bild 2: Ein Linienchart mit vielen zusätzlichen "Einbauten". Rechts unten die Toolbox.*

## Datenherkunft

Die Entwickler von S.&P.-CHART wissen, daß die Chartanalyse täglich "frische" Informationen braucht. So arbeiten sie an einer Anbindung des Programms an das Bildschirmtextsystem der Bundespost (BTX). Bis dies realisiert ist, kann im monatlichen Rhythmus eine Datendiskette für Aktien, Optionsscheine, Indizes und Währungen zusätzlich für DM 33,50 bestellt werden.

## Ein Fazit

Man merkt S.&P.-CHART an, daß die Entwickler selbst in der Anlageberatung tätig sind. Das Programm bietet Charts, die in der Börsenwelt üblich sind. Es zeichnet sich dadurch aus, daß alle gehandelten Werte (Aktien, Optionsscheine, Wärungen) und auch syntetische Werte (Branchen- und Länderindizes) grafisch dargestellt werden können. Außerdem unterscheidet sich S.&P.-Chart von der Konkurrenz durch die Möglichkeit, daß der Anwender sich selbst Kauf- und Verkaufskriterien herausarbeiten kann, um

# CW-CHART

## Der Tausendsassa



**Ein alter Hase des 'Parketts' soll auf die Frage, was er von technischer Analyse halte, wie folgt geantwortet haben: "Technische Analyse wurde durch Computer erst breitgetreten. Aber Computer sind nur in einer Form wirklich zu gebrauchen: als Aktie!"**

Eingefleischte 'Fundamentalisten' (also all jene, die sich um Kursgrafik, Linien und Balken keinen Deut scheren) werden wohl immer eine Art Kaffeesatzlesen in der Chartanalytik sehen. Als Fortsetzung der kleinen Reihe über Chart-Analyseprogramme für den ATARI ST soll im folgenden "CW-CHART" näher beschrieben werden.

Vor knapp einem halben Jahr hat das Software-Haus FOXWARE in Lenggries die Version 7.0 von CW-CHART fertiggestellt, welche uns zu diesem Test auch zur Verfügung stand. Die relativ hohe Versionsnummer läßt darauf schließen, daß das Programm einen langen Entwicklungsprozeß durchlaufen hat. Mal sehen, was dieses in der Scene doch recht bekannte Programm an Feinheiten zu bieten hat.

Um es gleich vorwegzunehmen - ohne Festplatte sollten Sie erst gar nicht beginnen, denn 1. sind die gelieferten drei Disketten (je nach Zusammenstellung) proppenvoll mit Dateien, und 2. wurden alle Verzeichnisnamen und Pfade für den Festplattenbetrieb (und dort nur auf Partition "D") im Programm-Code voreingestellt. Außerdem: Wenn Ihr ATARI weniger als 1 MByte Arbeitsspeicher aufweist, funktioniert CW-CHART leider auch nicht, gleiches gilt für einen Farbmonitor.

Es ist wirklich zu empfehlen, sich per Handbuch erst einmal einen Überblick von CW-CHART zu verschaffen, denn es besteht aus vier eigenständigen Programmen: 1. CHART, 2. DEPOT, 3. BTX und 4. TIMER. Wenn auch unser Hauptaugenmerk auf die Möglichkeiten der Analyse gerichtet bleiben soll, darf die Depotverwaltung nicht unerwähnt bleiben. Wie schon in früheren Berichten hervorgehoben wurde, bietet sich die "Nebenbei"-Verwaltung eigener Aktien durch ein Analyseprogramm geradezu an. Zu den Modulen "CW-BTX" und "CW-TIMER" später mehr.

Im Lieferumfang war auch ein kleines graues Kästchen, das als Kopierschutzstecker (engl. "Dongle") fungiert. Ich möchte meine ausführlichen Bemerkungen zu Kopierschutzmaßnahmen dieser Art nicht noch einmal wiederholen. Es sei mir hier lediglich die kurze Bemerkung erlaubt, daß ich für CW-CHART diese Art einer "Raubkopiererbremse" trotzdem für gerechtfertigt halte. Nun gut, wenn der "Dongle" schließlich im ROM-Port und alle Dateien auf Partition D Aufnahme gefunden haben, steht der Arbeit nichts mehr im Wege.

Der Eröffnungsbildschirm zeigt oben die übliche Menüleiste und am unteren Bildrand zwei Leisten für die Funktionstasten. Unter den Pull-Down-Menüs verbergen sich (auf dieser Programmebene) zahlreiche Verwaltungsaufgaben wie Kurseingabe, Aktien anlegen, Listen ausgeben usw. Es würde den Rahmen dieser Berichtserstattung sprengen, wenn all diese Feinheiten Erwähnung finden sollten. Für mein Gefühl sind die Verwaltungs-Pull-Downs wirklich ein wenig reichlich mit "Tätigkeiten" angefüllt, da hätte es doch sicher eine geschicktere Lösung gegeben.

Irgendwoher müssen die Basisdaten für unsere spätere Grafik ja kommen, und dazu bietet CW-CHART gleich zwei Wege an: 1. den bequemen über die "Automatik" per Bildschirmtext (BTX) oder 2. den mühsamen per Handeingabe. Nehmen wir der Einfachheit halber an, alle wichtigen Kurse lägen schon (elektronisch) vor; mit dem Menüpunkt "Grafik" kommen wir dann zum eigentlichen Mittelpunkt des Interesses:

### Die Chart-Palette

Jetzt wird die Stärke von CW-CHART richtig deutlich: In den Menüpunkten "Charts 1" und "Charts 2" sind sage und schreibe 33 verschiedene grafische Darstellungsarten untergebracht - das ist eine wahre Fundgrube für den Analytiker! So begegnen uns dort Begriffe (die in dem Grundlagenartikel von Heft 2/90 näher beschrieben wurden) wie Linie, Balken, Momentum, Point & Figure, High/Low. Aber auch eine Vielzahl von Tabellen ist hier zu finden.

Wenn die Entscheidung für eine Darstellungsart gefallen ist, z.B. Linien-Chart, erscheinen verschiedene Fragen für zusätzliche Parameter. Da CW-CHART mehrere Aktien in ein Bild projizieren kann, gilt die erste Frage der Anzahl der Aktien. Dann folgt der Name des Papiers (d.h. der Firma) oder dessen Kennummer. Per Maustaste kann man sogar in eine Liste wechseln und dort die Namen aussuchen. Auch der Start- und Endtag der Berechnung ist frei wählbar. Es schließt sich eine Frage nach der Anzahl von Tagen an, über die gewichtet bzw. gemittelt werden soll. Als letzte Einstellung

kommt die Frage nach dem Darstellungszeitraum. Diese Parameterzusammenstellung verläuft in zwei Menüleisten am oberen Bildrand, wobei in der obersten immer die momentan erlaubten Möglichkeiten aufgeführt sind. Nachdem das Chart-Bild erscheint, sind weitere Manipulationen machbar, wie Kommentare anfügen oder Formationen einzeichnen, z.B. Bogen, Trendkanal, Dreieck oder Viereck. Wirklich von Bedeu-tung sind die automatischen Kauf- oder Verkaufssignale, die CW-CHART beispielsweise in die TBI-, P&F- oder Parabolic-Tabelle setzt.

**Start**
Aktienordner ALT öffnen
Aktienordner öffnen
Aktienordner neu anlegen
Aktienordner Cutoff
Starte Programm
Einstellung sichern

**Kurseing.**
Alle Aktien
Mehrere Aktien
Eine Aktie
Liste Aktien
Liste Kurse
Kurse füllen
Kurse manipulieren
Kurslücken füllen
Extra

**Eingaben**
Neue Aktie hinzu
Aktie aendern
Aktie zeigen
Aktiensort
Aktie löschen
Währung löschen
Währung hinzu
Ereignis löschen
Ereignis ändern/hinzu
Editieren
Kurse -> Linkdatei
Linkdatei -> Kurse
Hol Quick
Mach Quick
Kennummerntest
Packs

**Listen**
Liste Aktien
Liste Datum
Liste Kurse
Liste Waehrungen
Liste Ereignisse
Druckliste Aktien
Druckliste Kurse

**Sonstiges**
Speicher räumen
Laufwerke anmelden
Directory
Konstanten
Diskplatz
Datum
Schreibmaschine
Drucker
Filelock
Ausgabeeinheit
File ALT sichern
File sichern
Datei löschen
Funktionstasten
Lerne
Stop Lernen
Lade CW-DEPOT
Ende

*Bild 1: Die Pull-Down-Menüs mit vielen organisatorischen Aufgaben*

## Von Editoren und Windows

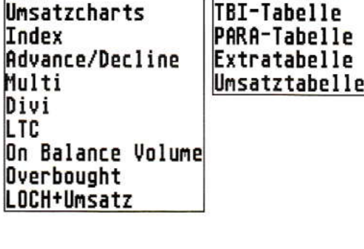
Nicht nur zu jedem neuen Chart werden diverse lokale Parameter abgefragt (d.h. nur für den folgenden Chart gültig - siehe vorhergehender Absatz), auch für allgemeinere Vorgänge sind (globale) Änderungen möglich. Diese betreffen dann alle weiteren Arbeiten.

Im "Moduseditor" legt man unter anderem die Art der Skalierung (linear oder logarithmisch), die Berechnung der Skala (automatisch oder manuell), das Linienmuster und die Rasterung fest.

Der "Fonteditor" bestimmt vornehmlich Größe und Typ der Textzeichen. Wenn ich ehrlich sein soll, geht die Definition für "Editor" (Schreibprogramm) etwas über das hinaus, was CW-CHART hierunter versteht, nämlich lediglich Auswahlfenster (aber was soll's?).

Ein höchst interessanter Punkt ist die "Bildlage". Damit kann die zu erstellende Grafik in einer frei wählbaren Größe auf dem Bildschirm positioniert werden. Da wir gerade bei der Bildschirmwiedergabe sind: CW-CHART arbeitet auf sechs verschiedenen Bildebenen bzw. Fenstern (engl. "Windows"), die alle auf einmal am Bildschirm sichtbar sein können, und für jedes ist die Bildlage unabhängig wählbar. Windows können einzeln gelöscht und sogar invertiert dargestellt werden.

Über den Menüpunkt "Konstanten" gelangt man in zwei Listen mit 39 Einstellgrößen, die ebenfalls Einfluß auf Darstellungen nehmen, ähnlich den sogenannten "Editoren". Lassen Sie mich beispielhaft nur ein paar nennen: Umkehrfaktor für Point&Figure, Chart-Breite in Tagen, Reihenfolge der Kurseingabe (Tag für Tag oder Aktie für Aktie), Kopfinformation in Chart einblenden usw.



**Charts 1**
Linechart
Balken
Line and Bar
Relativ
Vergleich
Trend
Momentum
Tagesschwankung
P+F
Hilo
Umsatzcharts
Index
Advance/Decline
Multi
Divi
LTC
On Balance Volume
Overbought
LOCH+Umsatz

**Charts 2**
RSI
Parabolic
Spread
Open Interest
Blättercharts
P+F-Tabelle
Stärketabelle
O-Scheintabelle
Gewinner/Verlierer
LTC-Tabelle
TBI-Tabelle
PARA-Tabelle
Extratabelle
Umsatztabelle

**Buffer**
Buffer
Bild
Ausschnitt
Cls
H/L

**Zeichnen**
Zeichnen
Tag
Wert
Text
Löschen
Scratchline
Letztes Bild

**Parameter**
Modusedit
Fontedit
Bildlage
Konstanten
Fadenkreuz
Hilfskreuz
Prozentomat
GD-Gewichtung
Window4
Cls Window
Chart>Window

*Bild 2: Im Analyseteil: Reichliche Auswahl zur Charterstellung*

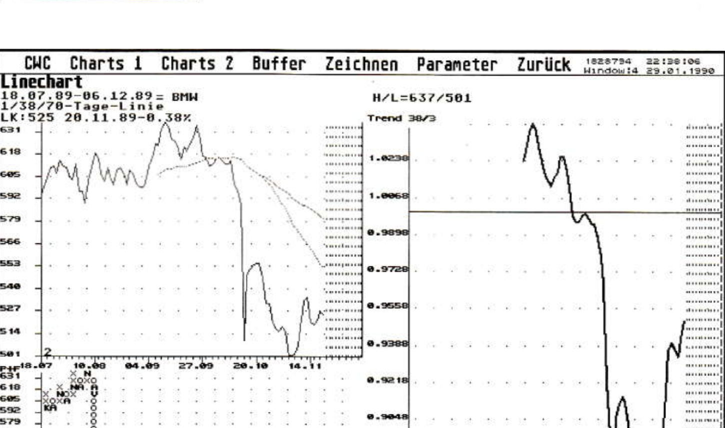



*Bild 3: Auch mehrere verschiedenartige Chartbilder sind auf einem Bildschirm darstellbar.*

Alles was man auf dem Bildschirm hat, möchte man natürlich irgendwann irgendwie festhalten - sprich abspeichern. Auch hier zeigt sich CW-CHART sehr kooperativ. Es stellt uns vier Bildschirmpuffer (engl. "Buffer") zur Verfügung, in die aus dem aktuellen Window zwischengespeichert, oder aus denen in das aktuelle Window geschrieben werden kann. Das "Schreiben" sollte man aber wörtlich nehmen, es ist nämlich ein "Überschreiben". Erst durch den weiteren Befehl "Bild -> Disk" landet der Inhalt eines Buffers oder Windows auf dem Externspeichermedium. Daß auch Ausschnitte entsprechend behandelt werden können, bedarf fast kaum noch der Erwähnung.

Eine Besonderheit ist die "Pack"-Funktion. Damit lassen sich verschiedene Aktien der gleichen Branche zur Berechnung eines eigenen Indexes verknüpfen. Damit läßt sich dieser Spezialindex (wenn z.B. nur Chemieaktien in die "Pack"-Datei kamen, dann wäre das ein individueller Chemieindex) hervorra-

gend mit einem allgemeinen Umsatzindex (z.B. "DAX-" oder "FAZ-Index") vergleichen.

CW-CHART ist sehr vielseitig und (zugegeben) an einigen wenigen Stellen etwas kompliziert. Macht aber gar nichts - denn: umfangreiche Befehlsabfolgen oder spezielle Einstellungen legt man einfach auf die 20 Funktionstasten (F11 bis F20 per SHIFT erreichbar). Die Belegung der Funktionstasten kann über zwei Wege stattfinden: entweder durch direktes Schreiben (wirklich wie eine Programmiersprache) oder im Lernmodus, wo die Abfolge der Anweisungen per Tastatur und Menüanwahl "vorgeführt" werden muß.

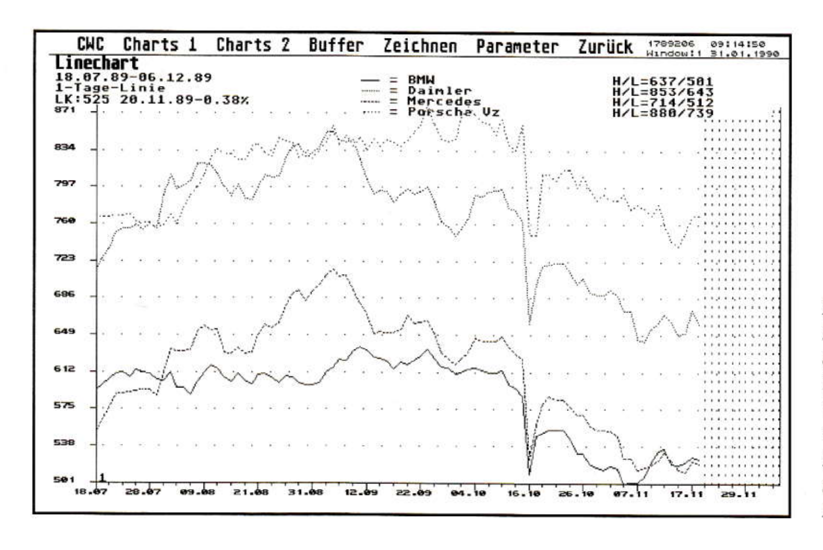



*Bild 4:* ***Sehr anschaulich ist der direkte Vergleich von (hier 4) Firmen der selben Branche.***

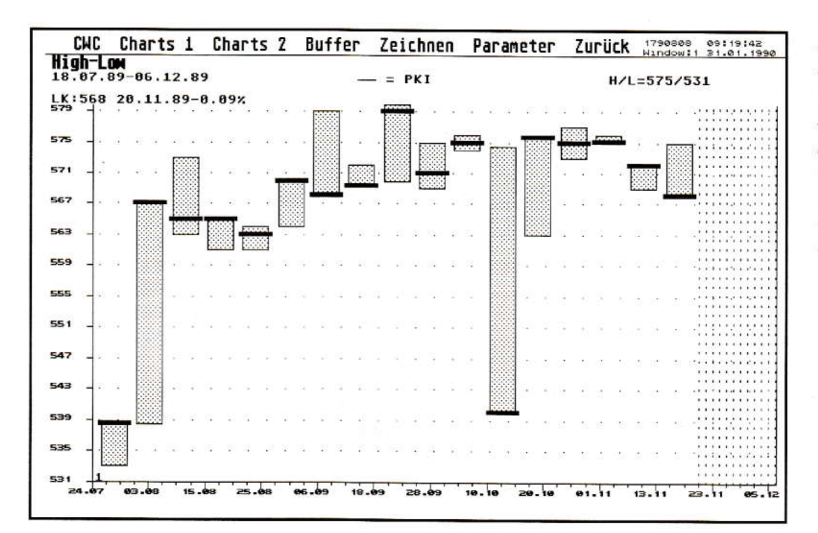



*Bild 5:* ***Die Bandbreite der Kursbewegungen wird bei Höchst/Tiefst-Balken sehr deutlich sichtbar.***

## BTX und TIMER

Es dürfte einleuchtend sein, daß bei einer einigermaßen sinnvollen Beschäftigung mit der Chart-Analyse ein Markt möglichst breit beobachtet werden muß. Das bedeutet, daß ich nicht nur meine eigenen Papiere, sondern die "Branchenverwandtschaft" ebenso wie Schlüsselindustrie und Indikatoren im Auge behalten soll. Damit der Überblick gewahrt bleibt, geht CW-CHART den Weg der Informationsbeschaffung per Bildschirmtext (Abk.: BTX). BTX hat den Riesenvorteil, daß es bei vergleichsweise niedrigen Kosten (DM 8,- monatliche Grundgebühr für ein Postmodem) bundesweit zum Ortstarif abrufbar ist. Und zudem bieten alle Großbanken und Sparkassen zuverlässig tägliche Kursnotierungen kostenlos an.

Für die Arbeit mit Bildschirmtext liegt ein Zusatzmodul namens "CW-BTX" bereit, das die Kursdaten aus den Angeboten der Kreditwirtschaft herausliest. Als Kommunikations-Software zwischen CW-BTX und dem Bildschirmtextsystem der POST fungiert das bekannte BTX-Programm "BTX-Manager" aus dem Hause Dieter Drews, Heidelberg.

Ein anderes Kommunikationsmodul ("HBL-BTX" und "HBL-TER") gibt es eigens für Verbindungen zu dem Brokerhaus "Hornblower Fischer" über BTX. Brokerhäuser bieten Kurse meistens nur in einer geschlossenen Benutzergruppe kostenpflichtig an, dann allerdings extrem schnell und mit vielen Zusatzinformationen. Mit jenen Anbietern muß für diesen Zweck ein Vertrag abgeschlossen werden.

Das Zeitsteuermodul "CW-TIMER" ist für jenen Fall vorgesehen, wenn die Kursabfrage per BTX unbeaufsichtigt (also vollautomatisch), auch täglich wiederkehrend, ablaufen soll. Es muß dann nur eine Schaltuhr für den Rechnerstart sorgen.

## Im Überblick

Gemessen an den "Einstiegsbedingungen" für CW-CHART (Festplatte, mindestens 1 MByte RAM, kein Farbmonitor) wird deutlich, an wen sich das Programm wendet. Es sollen hier bewußt Nägel mit Köpfen gemacht, also auf etwas angehobenem Niveau (auch preislich) gearbeitet werden. Das schlägt sich natürlich in einer Funktionsvielfalt nieder, die sicher unangefochten ihresgleichen sucht. So wie die reichhaltigen Ausstattungsmerkmale sehr erfreulich sind, drohen sie aber andererseits die Übersichtlichkeit zu erdrücken, besonders in den langen Pull-Down-Menüs. Die Handhabung von CW-CHART ist rundum angenehm und zufriedenstellend. Es gibt kaum Punkte, die man hätte anders oder besser gestalten können. Der scheinbar hohe Preis mag zunächst abschrecken, aber es werden dem Benutzer für 370 Papiere umgerechnet 100 000 historische Kursnotierungen mitgeliefert - und das tagesaktuell vom Versanddatum an rückwärts. Als hervorhebenswert gilt das Handbuch, welches auf 94 Seiten ausführlich und mit vielen Bildern in die Bedienung einführt. Speziell für den Spekulationsanfänger oder Kleinanleger gibt es eine "Junior"-Reihe derselben CW-Programme, nur mit geringfügig reduziertem Funktionsumfang. Eine Demoversion des Hauptprogramms ist ebenfalls verfügbar.

*DK*

***Preise:***

| | |
|---|---|
| *CW-CHART incl. CW-DEPOT* | *DM 997,-* |
| *CW-CHART junior* | *DM 399,-* |
| *CW-BTX* | *DM 199,-* |
| *CW-TIMER* | *DM 99,-* |
| *HBL-BTX* | *DM 399,-* |
| *HBL-TER* | *DM 399,-* |
| *Demoversion* | *DM 20,-* |

***Bezugsadresse:***

*FOXWARE*
*Buchsteinweg 1*
*8172 Lenggries*

# dBMan-Network

**Wenn man über Computernetzwerke spricht, meint man meistens ziemlich aufwendige und vor allem teure Steckkarten und Software. Daß es auch anders geht, zeigt die Firma Computer Mai in München mit ihrem Datenbankprogramm dBMAN.**

Dieses Programm ist an sich schon netzwerkfähig. Es beherrscht File- und Recordlocking, eine notwendige Einrichtung, um zu verhindern, daß zwei verschiedene Nutzer gleichzeitig denselben Datensatz verändern. Allerdings handelte es sich bei der getesteten Version von dBMAN-Network nicht um ein Netzwerk im herkömmlichen Sinne.



Vielmehr wurde hier für den ATARI die preisgünstigste Lösung eines "Netzwerkes" verwirklicht, mit einem großen Preisvorteil, aber auch vielen kleinen Fußangeln. Das Paket enthält neben einem extrem dünnen Handbuch drei Verbindungskabel für einen Masterrechner und zwei Datenstationen sowie je eine Diskette pro Rechner. Es lassen sich aber auch mehr Stationen in das Netz einbinden, wenn man die entsprechenden Kabel besorgt.

Das Handbuch beschränkt sich im wesentlichen auf die Erklärung und Demonstration der verwendeten Verfahren anhand von dBMAN-Quellcodeauszügen und eine Lizenzvereinbarung und bezieht sich in der zum Test vorliegenden Version hauptsächlich auf die alte Version 4 von dBMAN.

Die mitgelieferte Software besteht aus einer dokumentierten Beispielanwendung für das Netzwerk und einem kleinen Hilfsprogramm zur Vergrößerung des MIDI-Puffers. So, jetzt isses raus. dBMAN-Network verwendet den MIDI-Port des ST zur Vernetzung. Damit ergeben sich automatisch einige Probleme. In dBMAN, das für die Verwendung auf verschiedenen Rechnern vorgesehen ist, gibt es zwar Möglichkeiten, Daten über diese Schnittstelle zu senden und zu empfangen, aber die oben erwähnten Mechanismen zum Datenschutz sind nicht mehr wirksam. Außerdem ist die MIDI-Schnittstelle verhältnismäßig langsam, weniger als 40 kBaud im Vergleich zu bis zu 2.5 MBaud bei LANs wie z.B. Novell.

Bei dBMAN-Network handelt es sich, wie bereits erwähnt, nicht um die Netzwerkversion von dBMAN, die bisher nur für PCs und UNIX-Systeme verfügbar ist,

## Mit Netz und doppeltem Boden?

sondern vielmehr um eine Demonstration, wie man mit dBMAN eine akzeptable "Netzwerk"-Lösung erstellen kann. Der Trick beruht hierbei auf der Definition eines Protokolles, nach dem ein Master mit seinen Datenstationen Daten austauschen kann.

Entsprechend einem herkömmlichen Netzwerk können mehrere Benutzer die gleiche Datenbankdatei unter dBMAN nutzen. Diese befindet sich auf der Festplatte (die sollte es schon sein) des Masterrechners. Im Gegensatz zu normalen Netzwerken werden aber das Programm, nämlich dBMAN, und die Datenbankprozedur nicht vom Master in die Datenstationen geladen, sondern direkt auf diesen Stationen von Diskette.

Verbunden sind die Rechner über MIDI-Kabel zu einem Netzwerkring. Dabei ist es möglich, daß ein Rechner, der in den Ring eingebunden ist, nicht für die Datenbankanwendung verwendet wird. Es darf also z.B. eine Textverarbeitung laufen. Allerdings müssen alle Rechner im Netz eingeschaltet sein. Der Master muß als erster Rechner gestartet werden.

Die Rechner kommunizieren über ein Token-Verfahren, der Master schickt abwechselnd an jede Datenstation einen Abfragecode, um festzustellen ob eine Aufgabe vorliegt, und die angesprochenen Stationen schicken eine entsprechende Antwort. Das Paket enthält ein Anwendungsbeispiel, das ausreichend kommentiert ist, um es einem Programmierer zu ermöglichen, eine eigene Anwendung zu schreiben, die in diesem Netzwerk lauffähig ist.

Die Kosten für ein solches System sind relativ klein. Wie in den meisten Netzwerken braucht man einen 'dedicated server', was bedeutet, daß der Masterrechner, solange das Netz in Betrieb ist, nur für die Verwaltung des Netzbetriebes benötigt wird, und somit nicht zur Dateneingabe zur Verfügung steht. Ansonsten benötigt man für jeden ATARI, der an das Netz angeschlossen werden soll, nur ein zweiadriges Kabel, mit zwei 5-Pol-DIN-Steckern, wovon einer eine Diode enthält. Solche Kabel kosten nur einen Bruchteil der sonst üblichen Steckkarten oder Verbindungskästen.

Laut Computer MAI darf die Länge des Kabels zwischen zwei Arbeitsplätzen bis zu 50 Meter betragen, was in Büroräumen ausreichen sollte. Es gibt aber von anderen Herstellern Verstärker für die MIDI-Schnittstelle, die diese Reichweite um ein Vielfaches erhöhen können. Das allerdings hat dann seinen Preis.

Für wen lohnt sich nun ein solches Netz? Ich kann es mir gut in kleinen und mittleren Büros vorstellen, wo ein zentraler Datensatz für mehrere Mitarbeiter verfügbar sein muß und wo die Menge der übertragenen Daten 15-20 Bildschirmseiten pro Minute nicht als Dauerlast übersteigt. Das System wird inzwischen in verschiedenen Firmen eingesetzt, und bei Interesse läßt sich bestimmt einmal über eine solche Anwendung berichten.

*CSM*

***Bezugsadresse:***

*Computer Mai*
*Metzstr. 19*
*8000 München 80*
*Tel.: 089/4483820*

# FTL-Modula

## Die Feile, bitte!




**Das in Großbritannien schon geraume Zeit auf dem Markt befindliche FTL-Modula von Hi-Soft ist nun auch hier im Vertrieb von CCD erhältlich. Das System kostet DM 299,- und wird auf zwei doppelseitigen Disketten und mit einem Handbuch geliefert.**

### Installation

Vor der eigentlichen Installation müssen der Benutzername und die Seriennummer in die Systemprogramme eingetragen werden. Dazu dient das Programm *INSTALL*, in dem die nötigen Angaben getätigt werden. Damit hat CCD eine sinnvolle Lösung der Kopierschutzfrage gefunden, die auch bei ST-Pascal verwendet wird. Raubkopien lassen sich auf ihren Ursprung zurückführen, und wer nicht den korrekten Namen einträgt, ist von Updates ausgeschlossen. Dafür ist dann das komplette System frei von einem hinderlichen Kopierschutz. Das Testexemplar war von CCD übrigens schon vorinstalliert.

Die tatsächliche Installation beschränkt sich bei Systemen mit Festplatte auf das Kopieren sämtlicher Dateien der Systemdisketten in einen Ordner. Für die diversen Kombinationen aus Laufwerkskapazitäten und Speichergröße bietet das Handbuch verschiedenen Installationen an. Dabei werden hochformatierte Disketten und eine RAM-Disk verwendet. Nur die letzten Besitzer von einseitigen Diskettenlaufwerken müssen zunächst die Systemdisks bei CCD umtauschen.

Damit das System arbeiten kann, muß der Benutzer noch diverse Suchpfade und Systemoptionen über Environment-Variablen einstellen. Unter GEMDOS gibt es aber einige Probleme mit solchen Variablen, insbesondere lassen sie sich nicht in einem Profile oder einem *AUTOEXEC.BAT* setzen. Das Desktop bietet sie überhaupt nicht an, obwohl sie intern durchaus vorgesehen sind. FTL beinhaltet daher ein kleines Programm - *ENVIRON.PRG* -, das in den AUTO-Ordner kopiert wird. In einer zweiten Datei -*ENVIRON.DAT* - stehen die Inhalte der Variablen, die dann beim Booten automatisch gesetzt werden. Die Erzeugung von *ENVIRON.DAT* übernimmt der Batcher (s.u.).

Die Installation auf einer Festplatte belegt über 100 Dateien, für die ca. ein halbes Megabyte Plattenplatz nötig ist. Die Installation ist einfach und sollte aufgrund der genauen Beschreibung im Handbuch keinerlei Probleme aufwerfen.

### Oberfläche

Als Oberfläche bietet FTL keine der aufwendigen Grafik-Shells wie beispielsweise Megamax oder SPC. Vielmehr dient eine sehr einfache Shell, der "*BATCHER*", zur Steuerung. Dabei handelt es sich um einen Command-Line-Interpreter, der an das *COMMAND* von MS-DOS angelehnt programmiert wurde - und dementsprechend wenig bietet. Zur Dateipflege dienen die bekannten Operationen wie COPY, DEL, CD oder DIR.

Die schon angesprochenen Environment-Variablen verändert das SET-Kommando. Mit SAVE werden sie in die *ENVIRON.DAT*-Datei geschrieben und sind damit für das nächste Booten gerettet.

Etwas Komfort - beispielsweise gegenüber *COMMAND.TOS* von ATARI - bietet der Batcher bezüglich GEM-Programmen. Bei jedem gestarteten Programm mit der Endung PRG wird automatisch der Mauszeiger eingeschaltet, so daß keine Probleme bei Programmen mit grafischen Oberflächen entstehen.

Mag sein, daß die Arbeit mit einem Entwicklungssystem auf Shell-Basis vielen Programmierern entgegenkommt, deren persönlicher Arbeitsweise der Umgang mit der Maus etwas im Wege steht. Bei FTL allerdings ist die Shell zu einfach geraten, als daß sie wirklich einen Vorteil gegenüber den fulminanten Konkurrenzprodukten bietet. Es erscheint angeraten - und auch möglich -, den Batcher durch die Public Domain-Shell Guläm zu ersetzen. Neben der vorhandenen Alternative, die Systemprogramme vom Editor aufrufen zu können, ist eine einfache GEM-Shell in Vorbereitung. Die Arbeitsschritte werden dann mit Maus und Menüs unterstützt.

### Editor

Der GEM-gestützte Editor kann bis zu fünf Fenster verwalten und wird mit einer Mischung aus Tastatur- und Menükommandos (Bild 1) bedient. Die Tastatursteuerung lehnt sich stark an den Control-Tasten-lastigen Wordstar an, dessen Tastenbelegung auch heute noch in vielen Borland-Produkten zu finden ist. Für viele Befehle gibt es eine doppelte Belegung mit Alternate oder den Funktionstasten.

Vorhanden sind die üblichen Editor-Features mit Cursor-Bewegungen, Löschen, Suchen und Ersetzen sowie Blockbefehlen. Leider können Blöcke nicht mit der Maus markiert werden, zudem wird ein gesetzter Block nicht auf dem Bildschirm markiert.

Bis zu zehn Marken lassen sich setzen und anspringen. Findet der Compiler einen Fehler, kann weiter compiliert und die Fehlerposition in einer Marke vermerkt werden. Die Beschränkung auf nur zehn Marken und damit Fehler ist doch sehr knapp bemessen.

```
Datei
Öffnen
Sichern
Schließen
Fenster teilen
Datei verlassen
Fenster verlassen
Editor verlassen

Ausführen
Compiler
Überprüfen
Linker
Programm
Batcher

Optionen
✓ Dateisicherung
✓ Backups
✓ Warnton
  Zeilenumbruch
  Makro laden
  Compiler Flags

Tabs
  1 Spalte
  2 Spalte
  4 Spalte
✓ 8 Spalte
  AutoTab

Block
Blockanfang setzen
Blockende setzen
Block verschieben
Block kopieren
Block löschen
Zum Blockanfang
Zum Blockende

Marken
Marke 1 setzen
Marke 2 setzen
Marke 3 setzen
Marke 1
Marke 2
Marke 3
```

*Bild 1: Die Editor-Menüs*

Als einer der wenigen Editoren verwendet das Programm wirkliche Tabulatoren. Die Einrückungstiefe per Tabulator läßt sich im Menü *Tabs* einstellen. Ebenfalls dort findet sich die Option *AutoTab* zum automatischen Einrücken neuer Zeilen. Völlig unlogisch ist allerdings, daß eine Änderung der Tabulatorenweite das Auto-Indent jedesmal abschaltet.

Ein weiteres interessantes Feature ist die Möglichkeit, dieselbe Datei in zwei Fenstern darzustellen. Somit könnte man in einem Fenster eine Reihe Variablendeklarationen anzeigen und auf dem Bildschirm zur Erinnerung bereithalten, während in einem anderen Fenster eine Prozedur desselben Programms ausgearbeitet wird.

Für wiederkehrende Befehlsfolgen ist ein Makrorekorder implementiert. Nach *<ESC>,<Control>+L* kommt man in einem Lernmodus, der sämtliche Aktionen mitschreibt und nach einem erneuten *<ESC>,<Control>+L* bereitstellt. Die Makros werden durch beliebige Zeichenketten bezeichnet, die beim Schreiben automatisch gesetzt werden. Verwendet man aber längere Namen, hat das zur Folge, daß der Editor die Eingabe verzögert übernimmt, da sie erst mit den bekannten Makro-Namen verglichen wird. Vielleicht sollte ein Makro-Aufruf doch durch einen Befehl eingeleitet werden, wenn der Name mehrere Zeichen umfaßt.

Durch die Makros wird es möglich, die komplette Tastenbelegung des Editors umzudefinieren. In der Auslieferung befinden sich drei Belegungsdefinitionen, die jeweils mit *Makro laden* eingestellt werden können. Diese Bindings werden in Klartext definiert; eine konkrete Anleitung dazu findet sich aber nur in den - durchaus gut kommentierten - mitgelieferten Dateien.

Der Editor hat leider einige Probleme mit GEM. Die Reaktionszeit auf Slider-Bewegungen ist ungewohnt hoch. Die Größe der Fenster wird bei Bewegungen automatisch an die Bildschirmränder angepaßt. Es ist also nicht möglich, ein Fenster aus dem Bildschirm teilweise herauszuschieben. Die Folge wird eine größere Zahl von Sizer-Operationen durch den Benutzer sein.

Der Compiler läßt sich im Menü *Ausführen* mit und ohne Code-Erzeugung starten; ebenso der Linker. Der Menüpunkt *Programm* führt ein beliebiges Programm aus, womit alle anderen System- und Hilfsprogramme bereitstehen. Die Shell kann mit *Batcher* gestartet werden. *Compiler Flags* im Menü *Optionen* erlaubt das Anwählen der Compiler-Optionen. Seltsamerweise fehlt ein entsprechendes Kommando für den Linker.

Neben leichten Problemen mit GEM müßte der Editor überarbeitet werden, soll er zugleich die Entwicklungsoberfläche integrieren. So müssen noch einige Feinheiten bereinigt und etwas mehr Komfort geboten werden. Der Editor hat also noch etwas Überarbeitung nötig. Solange sollte die Verwendung eines anderen Editors problemlos sein, besondere Probleme bei der Einbindung beispielsweise von TEMPUS dürften nicht entstehen.

## Compiler

Der FTL-Compiler erweitert Modula-2 an einigen Stellen. So dürfen in Bezeichnern beispielsweise auch die Zeichen "$" und "_" vorkommen.

Teilweise implementiert ist die Möglichkeit - ähnlich wie in C -, Variablen bei ihrer Deklaration einen Initialwert zuzuweisen. Zeichenkonstanten können neben der üblichen oktalen Darstellung auch im hexadezimalen Zahlensystem notiert werden. Opaque Type müssen nicht mehr unbedingt vom Typ POINTER sein. Dazu ändert der Compiler beim Übersetzen des Implementationsmoduls nachträglich die .SYM-Datei.

Prozedurkonstanten lassen Zeilen wie "CONST Schreibe = WriteString" zu. Im Programm wird dann *Schreibe* jeweils ersetzt, und zur Verwendung einer anderen Ausgabeprozedur muß lediglich die Konstantendefinition geändert werden. Anstelle einer direkt geschachtelten WITH-Anweisung kann auch in einem WITH eine Liste von Record-Variablen stehen.

Ein weiterer Unterschied, den ich eher als Fehler bezeichnen möchte, ist das obligatorische BEGIN in einem Prozedurkörper. Eine Deklaration mit einem leeren Prozedurkörper *PROCEDURE a; END a;* ist nicht mehr möglich, es muß *PROCEDURE a; BEGIN END a;* lauten.

Jede Benutzung dieser Erweiterungen macht die Programme natürlich nicht mehr portabel. Ob sie bei dieser Einschränkung allerdings noch Vorteile bringen, ist fraglich. Es gibt eigentlich keinen Grund, das Dollar-Zeichen und den Unterstrich verwenden zu müssen, ebenso lassen sich die etwas unsichereren Prozedurkonstanten (keine Festlegung der Parameterstrukturen) auch über Prozedurvariablen realisieren. Auf jeden Fall aber wird FTL nicht sofort an den neuen Modula-Standard anzupassen sein, sollte dieser eines Tages tatsächlich verabschiedet werden.

Der Compiler wird entweder vom Editor oder aus dem Batcher unter Angabe des Quelltexts aufgerufen. Dabei können verschiedene Optionen gesetzt bzw. nach einem Schrägstrich an den Programmaufruf angehängt werden.

Die Optionen steuern das Verhalten des Compilers beim Übersetzen. Dabei können ein Bildschirm-Listing ausgegeben sowie Warnungen ausgeschaltet werden. Die Code-Erzeugung beeinflußt die bekannten Schalter zur Eincompilierung von Code zur Bereichsüberprüfung bei Feldzugriffen sowie der Erkennung von Overflow-Fehlern. Diverse weitere Schalter steuern die Erzeugung von Zusatzinformationen für das Debugging.

# GAL-Prommer




## MGP - Einfach, schnell und praktisch

Die Zeiten der elektronischen Schaltungen, als Röhren mit ihrer Heizwendel oder gar Relais viel Strom und Platz verbrauchten, sind längst vorbei. Moderne Computerbausteine und Logik-Chips finden sich in fast jeder Schaltung wieder. Kein Fernseher, keine Aufzugssteuerung und schon gar keine Computerschaltung kommt ohne Steuerlogik und Verknüpfungsbausteine aus.

Aber auch die Zeiten, in denen man mit unzähligen sogenannter TTL-Gräber oder CMOS-Bausteinen eine Schaltung entwickelte, gehen vorüber. Nach den frei programmierbaren, aber nicht löschbaren PAL-Bausteinen werden immer häufiger die frei programmierbaren und löschbaren und somit viel flexibleren GAL-Bausteine eingesetzt. Kein Wunder, denn durch ein einziges GAL lassen sich viele der herkömmlichen Logikbausteine ersetzen.

Neben der enormen Platz- und Stromersparnis bleibt eine einmal aufgebaute Schaltung flexibel, denn ein GAL ist in wenigen Sekunden umprogrammiert und kann somit neuen Anforderungen angepaßt werden. Typische Einsatzgebiete eines GALs sind:

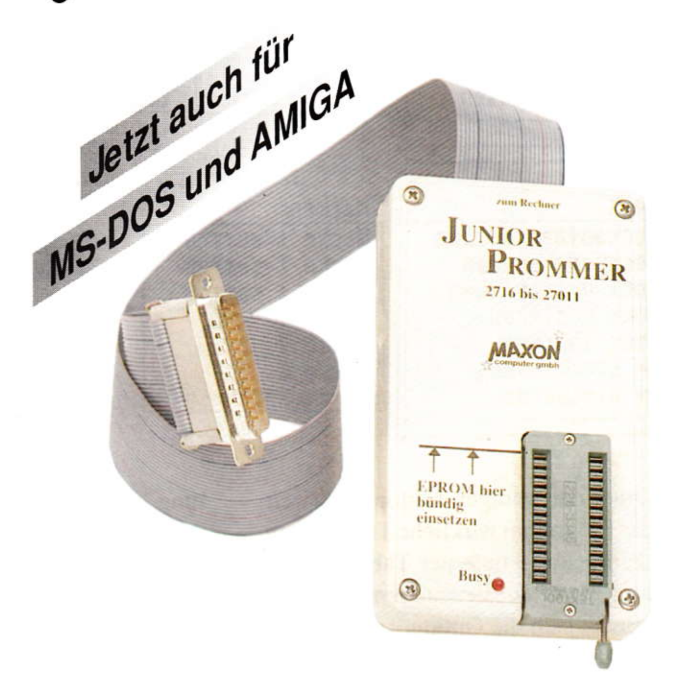
1. **Adreßdekoder**
2. **Zustandsautomaten**
3. **logische Gatter**
4. **PAL-Simulation**

Der MAXON GAL- Prommer 16/20 bietet Ihnen alle Möglichkeiten, die bekannten GAL-Typen 16V8 und 20V8 mühelos zu programmieren. Nutzen Sie diese modernen Bausteine, um Ihre Schaltungen schnell, preiswert und flexibel aufzubauen. Mit dem MGP 16/20 kein Problem, denn im ausführlichen Bedienungshandbuch wird auch erklärt, wie herkömmliche Logik (NOR-, NAND-, NOT-, ...-Gatter) in ein GAL programmiert wird und wie diese Bausteine arbeiten.

Der MGP 16/20 läuft an jedem ATARI ST und wird an die Druckerschnittstelle angeschlossen. Die benötigte Betriebsspannung wird am Joystick-Port abgenommen. Die menügesteuerte Software ermöglicht ein bequemes und sicheres Arbeiten mit den GALs. Im eingebauten Editor läßt sich die JEDEC-Datei (so heißt die Datei, die in ein GAL programmiert wird) leicht erstellen oder ändern.

**Ein Muß für jeden Elektroniker - GALs, die neue Generation der programmierbaren Logik!**

# Junior Prommer




## Klein, kompakt und leistungsstark

Der Junior Prommer programmiert alle gängigen EPROM-Typen, angefangen vom 2716 (2 kByte) bis zum modernen 27011 (1 MBit). Aber nicht nur EPROMs, sondern auch einige ROM- und EEPROM-Typen lassen sich lesen bzw. programmieren. Zum Betrieb benötigt der Junior Prommer nur +5 Volt, die am Joystick-Port Ihres ATARI ST abgenommen werden, alle anderen Spannungen erzeugt die Elektronik des Junior Prommers. Selbstverständlich läßt sich ein 16-Bit Word in ein High- und Low-Byte zerlegen. Fünf Programmieralgorithmen sorgen bei jedem EPROM-Typ für hohe Datensicherheit. Im eingebauten Hex/ASCII-Monitor läßt sich der Inhalt eines EPROMs blitzschnell durchsuchen oder auch ändern.

Bemerkenswert ist der Lieferumfang, so wird z.B. das Fertiggerät komplett aufgebaut und geprüft im Gehäuse mit allen Kabeln anschlußfertig geliefert. Auf der Diskette mit der Treibersoftware befinden sich noch RAM-Disk und ein Programm zum Erstellen von EPROM-Karten, ferner wird der Source-Code für Lese- bzw. Programmierroutinen mitgeliefert und last but not least ist im Bedienungshandbuch der Schaltplan abgedruckt.

| Typ | | | |
|---|---|---|---|
| INT. ID. | | | |
| EPROM'S | | | |
| 2716 | 25V | | |
| 2732 | 25V | 2732A | 21V |
| 2764 | 21V | 2764A | 12V |
| 27128 | 21V | 27128A | 12V |
| 27256 | 12V | 27256 | 21V |
| 27512 | 12V | 27513 | 12V |
| 27011 | 12V | | |
| ROM'S | | | |
| 4732 | | 4764 | |
| 47128 | | 47256 | |
| EEPROM'S | | | |
| X2804A | | X2816A | |
| X2864A | | X28256A | |

***Übersicht der mit dem Junior Prommer programmierbaren Speichertypen***

Mit den bekannten Pseudokommentaren können die Einstellungen auch selektiv für bestimmte Bereiche des Programmtextes ein- und ausgeschaltet werden.

Hier kommt eine weitere Option hinzu, die nicht beim Compiler-Aufruf gesetzt werden kann. In String-Konstanten lassen sich unter Verwendung eines Escapes auch nicht-druckbare Zeichen darstellen. So erzeugt beispielsweise "^L" ein Form-Feed (ASCII-Code 12). Mit (*$A!*) würde das voreingestellte Escape "^" auf das Ausrufezeichen umgestellt.

## Benchmarks

In Bild 2 finden Sie die gewohnte Benchmark-Tabelle, in der FTL nun schon die achte Spalte füllt. Die ersten Ergebnisse (Benchmarks 2-5) zeigen, daß FTL der erste Compiler ist, der aus einem i:=i+1 ein INC(i) optimiert.

Dagegen erreicht FTL bei den verschiedenen Kontrollkonstrukten (8-10, 21-24) wie Schleifen oder Fallunterscheidungen nur mittelmäßige Werte. Hier scheint die Konkurrenz etwas besser auf den 68000 zu optimieren.

Da REAL mit 64 Bit implementiert ist, wurden die Zeiten den LONGREAL-Ergebnissen der Konkurrenten gegenübergestellt (Benchmark 26). Es zeigt sich, daß die Fließkomma-Arithmetik nicht optimal vorliegt, der Benchmark lieferte das schlechteste Ergebnis. Beim Test der Libraries (27a) schlägt dieser Wert erstaunlicherweise nicht zu stark durch, dennoch sind die höheren REAL-Funktionen ebenfalls nicht sonderlich schnell. Ein Blick in den Quellcode der *MathLib* zeigt, daß alle Aufrufe an weitere Module durchgereicht werden, und dort unoptimiert in Hochsprache formuliert sind.

Trotz der schnellen Feldzugriffe (29) konnte sich auch die String-Verarbeitung keinen Spitzenplatz erobern (28). Insgesamt scheint FTL Stärken in der Ganzzahl-Arithmetik zu haben, verfehlt bei den Kontrollkonstrukten aber deutlich die Zeiten der Konkurrenz. Die Bibliotheken sind nicht sonderlich optimiert, eine Aufgabe, die hoffentlich nicht auf die Benutzer abgewälzt bleibt.

**RT Modula-2 Benchmarks**

| | | Jefferson | | Softwave | | LPR | | FTL | |
|---|---|---|---|---|---|---|---|---|---|
| Nr. | TDI V3.0 | | Megamax | | SPC V1.41 | | MAMOS V1.3 | | testet ... |
| 1 | 0:07 | 0:07 | 0:07 | 0:04 | 0:05 | 0:05 | 0:02 | 0:04 | Prozeduraufruf |
| 2 | 1:42 | 1:33 | 2:59 | 1:35 | 1:31 | 1:31 | 1:27 | 1:17 | Addition |
| 3 | 1:21 | 1:18 | 1:58 | 1:20 | 1:16 | 1:16 | 1:12 | 1:17 | Increment |
| 4 | 1:47 | 1:38 | 2:59 | 1:40 | 1:36 | 1:36 | 1:32 | 1:22 | Additionsoptimierung |
| 5 | 1:27 | 1:23 | 2:08 | 1:25 | 1:21 | 1:22 | 1:17 | 1:22 | Increment als Vergleich |
| 6 | 2:09 | 1:57 | 3:48 | 2:01 | 1:55 | 1:55 | 1:51 | 1:37 | INTEGER-Addition |
| 7 | 2:09 | 1:57 | 3:48 | 2:01 | 1:55 | 1:55 | 1:51 | 1:37 | CARDINAL-Addition |
| 8 | 1:11 | 1:18 | 1:42 | 1:05 | 1:15 | 1:16 | 1:12 | 1:22 | FOR-Schleife |
| 9 | 1:21 | 1:02 | 1:42 | 1:05 | 1:00 | 1:00 | 0:56 | 1:02 | REPEAT-Schleife |
| 10 | 1:21 | 1:18 | 1:57 | 1:20 | 1:15 | 1:15 | 1:12 | 1:17 | WHILE-Schleife |
| 11 | 1:04 | 0:54 | 1:15 | 0:38 | 0:51 | 0:53 | 0:48 | 0:45 | INTEGER-Parameter |
| 12 | 1:04 | 0:54 | 1:17 | 0:38 | 0:51 | 0:53 | 0:48 | 0:44 | INTEGER VAR-Parameter |
| 13 | 1:06 | 0:59 | 2:19 | 0:33 | 0:57 | 0:57 | 0:53 | 0:58 | RECORD-Parameter |
| 14 | 0:34 | 0:30 | 0:41 | 0:20 | 0:28 | 0:28 | 0:24 | 0:25 | RECORD VAR-Parameter |
| 15 | 0:49 | 0:49 | 1:33 | 0:42 | 0:46 | 0:47 | 0:43 | 0:44 | Konstanten-Optimierung |
| 16 | 0:51 | 0:51 | 1:33 | 0:44 | 0:49 | 0:49 | 0:45 | 0:52 | Konstanten-Optimierung |
| 17 | 1:28 | 1:26 | 2:06 | 1:19 | 1:23 | 1:24 | 1:20 | 1:28 | Expression-Optimierung |
| 18 | 1:42 | 1:22 | 1:59 | 1:15 | 1:19 | 1:20 | 1:16 | 1:23 | Expression-Optimierung |
| 19 | 0:37 | 0:36 | 0:55 | 0:30 | 0:33 | 0:34 | 0:29 | 0:33 | Zwischenergebnis-Optimierung |
| 20 | 0:37 | 0:35 | 0:55 | 0:30 | 0:33 | 0:34 | 0:29 | 0:33 | Zwischenergebnis-Optimierung |
| 21 | 0:09 | 0:11 | 0:13 | 0:09 | 0:08 | 0:09 | 0:05 | 0:10 | IF-Statement |
| 22 | 0:13 | 0:13 | 0:16 | 0:11 | 0:11 | 0:12 | 0:07 | 0:12 | IF durch CASE ausgedrückt |
| 23 | 0:38 | 0:33 | 0:41 | 0:28 | 0:30 | 0:31 | 0:27 | 0:38 | CASE-Statement |
| 24 | 0:40 | 0:39 | 1:03 | 0:38 | 0:36 | 0:36 | 0:33 | 0:37 | CASE durch IF ausgedrückt |
| 25 | 0:47 | 1:03 | — | 2:09 | 0:49 | 0:48 | 0:42 | — | REAL-Arithmetik |
| 26 | 2:05 | — | 1:32 | 2:18 | 2:07 | 2:00 | 1:58 | 2:37 | LONGREAL-Arithmetik |
| 27 | 1:52 | 5:42 | — | — | 3:51 | 3:16 | — | — | REAL-Library |
| 27a | 5:39 | — | 2:35 | 35:40 | 3:30 | — | 16:13 | 6:28 | LONGREAL-Library |
| 28 | 1:21 | 1:21 | 0:40 | 0:30 | 0:51 | — | 2:20 | 1:28 | String-Library |
| 29 | 2:10 | 2:07 | 2:13 | 1:48 | 1:44 | 2:04 | 2:01 | 1:33 | ARRAY-Zugriffe |
| 30 | 0:09 | 0:10 | 0:17 | 0:19 | 0:08 | 0:08 | 0:04 | 0:12 | RECORD-Zugriffe |

Alle Zeiten mit time-Kommando von Guläm gemessen
MAMOS 1.3 mit 200Hz-Zähler gemessen
Meßgenauigkeit bis zu ±0.5 Sekunden
Angabe "'—"': Sinnlos bzw. keine Bibliotheken

*Bild 2: Die Benchmarks*

## Linker

Der Linker (Bild 3) bindet die vom Compiler oder Assembler (s.u.) erzeugten Link-Dateien zu einem GEMDOS-Programm zusammen. Über einen Schalter lassen sich auch Accessories erzeugen.

Zur Optimierung des gebundenen Programms, also der Entfernung aller nicht benötigten Routinen, ist ein etwas umständliches Procedere nötig. Zunächst muß ein Link-Vorgang durchgeführt werden, der über ein Flag eine Datei mit Informationen über die Prozeduraufrufe erzeugt. Diese wird durch ein weiteres Programm - den TRIMMER - geschickt, der wiederum eine Informationsdatei für den Linker erzeugt, die beschreibt, an welchen Stellen Code herausgelassen werden kann. Der Linker verarbeitet diese dann in einem zweiten Link-Vorgang und erzeugt endlich ein kompakteres Programm. Die Konkurrenz zeigt, daß diese Arbeitsschritte - die allerdings nur für die endgültige Programmversion nötig sind - auch direkt im Linker integrierbar sind und nicht etwa einen doppelten Linkeraufruf und die Verwendung eines weiteren Programm bedingen.

Weitere Flags steuern Stack-Überprüfung und -größe und können die vom Compiler erzeugte Bereichsprüfung wieder rückgängig machen. Wie beim Compiler gibt es auch hier einige Flags, die das Programm durch Zusatzinformationen zum Debuggen vorbereiten. Schließlich läßt sich auch ein Mathe-Coprozessor durch Einbinden entsprechender Module ansteuern.

## Bibliotheken

Die externen Module liegen bei FTL-Modula in Sammelbibliotheken vor. Eine solche Bibliothek hat ein eigenes Inhaltsverzeichnis und faßt die vielen sonst üblichen kleinen Einzeldateien in einer großen zusammen. Die Dateien können dabei mit zwei verschiedenen Komprimierungsarten gepackt werden. Zur Verwaltung steht ein Bibliotheksmanager bereit, der weiter unten besprochen wird. Der Vorteil des Verfahrens liegt auf der Hand. Anstelle von hunderten von Einzeldateien wird nur noch mit wenigen Sammelbibliotheken gearbeitet. Vorrangig spart dieses Verfahren Platz und ist eventuell auch schneller, da nicht mehr immer ein GEMDOS-Open durchgeführt werden muß. Eine zusätzliche Bibliothek enthält übrigens den Quellcode sehr vieler mitgelieferter Module.

```
Erstelle: G:\FTL\ALLQUEEN.PRG
ALLQUEEN     6     0    52 INOUT        174     0    A8 SYSTEM      DDE     4    AC
TERMINAL  12E8    62   140 CONVERSI    16B6    66   154 STREAMS    1896    76   2B0
BIOS      276E    82   2B8 STORAGE     2DB8    82   2D2 GEMDOS     34FC    8A   2D2
LONGINTL  36D4    8A   2D2


Hauptprogramm wird zusammengebunden
TERMINAL   16A6 STORAGE    34C0 STREAMS    2736
INOUT       D3C ALLQUEEN    102
Daten  - Segmentlänge :   2D2
Text   - Segmentlänge :  3768
BSS    - Segmentlänge :    8A                 [Weiter]

Linken erfolgreich
```

*Bild 3: Ein Linker-Lauf*

Die Modula-Standardmodule wie *Storage* oder *Strings* sind vorhanden. Die Ein- und Ausgabe wird neben *InOut* abgestuft angeboten. Dabei existiert ein "höheres" Modul - *ScreenIO* -, das Textfenster mit Attributen verwalten kann. Die Strukturierung der Bibliotheken lehnt sich wenig an den empfohlenen Standard an, so heißt *MathLib0* bei FTL *Maths*.

An vielen Stellen zeigt sich der Ursprung der Bibliotheken aus MS-DOS. So sind einige Module zur Interrupt-Verwaltung vorhanden; ein Bereich, der auf dem PC ungleich wichtiger ist.

An Besonderheiten gibt es beispielsweise eine Bibliothek zum Lösen von Gleichungssystemen oder einen Quicksort. Ein paar sehr systemnahe Module erlauben das schnelle Verschieben von Speicherblöcken und Bit-Manipulationen.

ATARI- bzw. GEM-spezifische Module sind im üblichen Umfang vorhanden. Dies reicht von BIOS, XBIOS, GEMDOS und der Line-A bis zu den gewohnten AES- und VDI-Bibliotheken. Die Bezeichner bei letzteren entsprechen in etwa den Vorlagen aus dem C-Entwicklungspaket von ATARI. Damit sind sie kryptisch, schlecht zu lesen und zu merken und nutzen Modula nicht aus. Höhere Module für die GEM-Benutzung sind nicht in dem Umfang wie bei der Konkurrenz zu finden.

Die Beurteilung der Bibliotheken ist schwierig. Eigentlich ist alles vorhanden, was man auf dem ST braucht. Im Vergleich mit der Konkurrenz aber fehlen die aufwendigen Komfort-Module für GEM, und es gibt eigentlich keine Stelle, an der die FTL-Module Besonderes bieten.

## Assembler

Dem Paket liegt ein 68000-Assembler mit Schnittstelle zum Modula-System bei. Die Autoren begründen dies damit, daß einige der mitgelieferten Bibliotheksquellen in Assembler vorliegen, und weisen ausdrücklich darauf hin, daß das Programm keine vollständige Assembler-Umgebung bietet. Die Aussage im Handbuch, "Außerdem waren wir beim Testen des Assemblers ... nicht so gründlich wie bei den anderen Komponenten des Systems - er arbeitet jedoch zufriedenstellend", klingt wie eine Entschuldigung und verunsichert den Anwender.

Der Assembler-Text wird mit Motorola-Mnemonics in der üblichen Notation geschrieben. Vordefiniert sind Bezeichner für die Register und einige Operatoren für Ausdrücke. Die Assembler-Pseudoanweisungen beinhalten das Nötigste, wie DB, DW, DW, EVEN oder EQU. SET kann mit veränderlichen Marken arbeiten.

Ein Assembler-Modul besteht aus einem in Modula geschriebenen Definitionsmodul und einem Assembler-Text, der praktisch das Implementationsmodul darstellt. Der Assembler erzeugt daraus ein Linkmodul wie der Compiler.

Der Assembler bietet eine Schnittstelle, mit der Bezeichner im- und exportiert werden können. Dabei finden allerdings keine Typüberprüfungen statt, womit bei den ersten Versuchen wahrscheinlich einige schwer zu findende Fehler auftreten.

Eine Pseudo-Anweisung - IMPORT - gefolgt von einem Modulnamen und einer Namensliste importiert Bezeichner aus externen Modulen. Die Anweisung LABEL gibt einen Bezeichner nach außen bekannt und wird somit jede in Assembler definierte Prozedur einleiten.

Das Handbuch beschreibt die einfache Prozeduraufruf-Konvention. Daraus ein Beispiel zur Ausgabe eines Textes auf dem Bildschirm:

```
    IMPORT Terminal, WriteString, WriteLn
    ISECT
Hi: DB Hi There,0
    CSECT
    MAIN
    PEA     Hi
    MOVE.L  #8,-(A7)
    MOVE.L  #7,-(A7)
    JSR     WriteString
    JSR     WriteLn
    RTS
    END
```

Nach dem Import von *WriteString* und *WriteLn* aus *Terminal* folgt ein Datenbereich, der den auszugebenden String definiert. Die Vorbereitung des Aufrufs von *WriteString* besteht aus dem Ablegen der String-Adresse, der Anzahl der belegten Bytes und der oberen Feldgrenze. Die letzten beiden Werte sind nötig, da *WriteString* ein offenes Feld als Parameter erwartet. Der tatsächliche Aufruf von WriteString geschieht über ein simples JSR.

Assembler wird von den Autoren nur zur Geschwindigkeitsoptimierung empfohlen. Größere Programme in Assembler zu entwickeln wird wahrscheinlich auch an dem mangelnden Komfort scheitern. Gerade Prozeduraufrufe könnten Makros erheblich vereinfachen.

Der Assembler ist also eher ein Hilfsprogramm für letzte Optimierungen, das aber bei der normalen Programmentwicklung eine eher untergeordnete Rolle spielen wird. Gegenüber anderen Lösungen mit Inline-Assembler paßt sich die Erzeugung von Implementations-Modulen durch den separaten Assembler besser in das Modula-Konzept ein.

## Debugger

Zum Debuggen existiert ein symbolischer Debugger, der Einzelschrittverarbeitung auf Anweisungsniveau, Variablen-Monitoring und Break-Punkte bietet.

Das Programm - das auch nicht auflösungsunabhängig ist - meldet sich auf einem eigenen Bildschirm mit einem eigenen Menü (Bild 4). Nachdem ein Modul zur Abarbeitung geöffnet ist (dies ist normalerweise das Hauptmodul), lassen sich einige Einstellungen zum Entwanzen vornehmen.

"Var anzeigen" nimmt eine (per Menü ausgewählte) Variable in die ständige Anzeige auf. Will man nur kurz über den momentanen Wert einer Variablen informiert sein, wird er mit "Var untersuchen" temporär dargestellt. Für die Darstellung gibt es verschiedene Optionen, so kann

beispielsweise ein Feld ab einem beliebigen Index angezeigt werden.

Jeweils eine Anweisung führt "Nächster Schritt" aus. Dabei sind aber nicht etwa 68000-Anweisungen, sondern Modula-Anweisungen gemeint. Die Schrittweite läßt sich ebenfalls einstellen, so daß man beispielsweise nur nach jeder dritten Anweisung wieder im Debugger landet. "Nächste Anweisung" führt das Programm bis zur nächsten Anweisung auf gleichem Niveau aus. Damit gelten ein Prozeduraufruf oder eine Schleife als eine Anweisung.

Break-Punkte lassen sich ebenfalls durch Angabe einer Zeilennummer setzen und löschen. "Ausführen" arbeitet das Programm danach solange ab, bis es auf einem Break-Punkt stoppt oder ein Laufzeitfehler auftritt. Alle Ausgaben eines Programms gehen auf einen anderen Bildschirm, der sich mit "PRG Bildschirm" anzeigen läßt.

"Zeit-Modul" fertigt ein Zeitprofil des Programmablaufs an. In einer dabei entstehenden Datei ist danach vermerkt, welche Anweisung wieviel Zeit des gesamten Programmlaufs beansprucht. Damit lassen sich sehr gut kritische Stellen im Programm aufspüren und Optimierungen zielgerichtet durchführen.

Die nicht direkt lesbare Datei mit dem Zeitprofil wird danach vom Programm PRTIME umgesetzt. Nach Auswahl in einer Dialogbox lassen sich die Laufzeiten (in Anteilen an der gesamten Programmdauer) auf dem Bildschirm (Bild 5) oder in einer externen Datei darstellen. Bei letzterem fehlen leider unverständlicherweise die Zeitangaben. Die Bildschirmdarstellung hat einige Schwächen bei der GEM-Benutzung.

Beim Austesten des Debuggers traten einige Ungereimtheiten auf. Bei einem gesetzten Break-Punkt wurde nur einmal gestoppt, obwohl er sich in einer Schleife befand. Das Zeitprofil konnte nicht befriedigen (Schleifen im Hauptteil eines Programms wurden nicht ausgemessen) und bei der Ausgabe in einer Datei wurden die Einrückungen plötzlich nicht mehr beachtet, bei einem Durchlauf brach die Darstellung willkürlich im Programm ab. Der Debugger bzw. das Zeitprofil scheinen noch fehlerhaft implementiert zu sein. Das Konzept ist zwar das richtige, aber durch die Fehler wird die Nutzbarkeit des Werkzeugs an vielen Stellen eingeschränkt.

```
PC=   87A46 Flags = 300
D0=       7 D1=       1 D2=       0 D3=       0
D4=       0 D5=       0 D6=       0 D7=       0
A0=      C8 A1=   8B0B8 A2=       0 A3=   8B5D4
A4=       0 A5=       0 A6=       0 A7=   ABC30

  87A46 Zeile   36 ALLQUEEN
  8B5D4 i                 7
  8B5FE x              [   0,    0,    0,    0,
  0,    0,    0,    0 ]
  8B5DE b              [ FALSE , FALSE ,
FALSE , FALSE , FALSE , FALSE , FALSE , FALSE ,
FALSE , FALSE  ]
  8B5D6 a              [ TRUE , TRUE , TRUE ,
TRUE , TRUE , TRUE , FALSE , FALSE  ]
  8B5EE c              [ FALSE , FALSE ,
FALSE , FALSE , FALSE , FALSE , FALSE , FALSE ,
FALSE , FALSE  ]

Nächster Schritt  F1
Modul öffnen      F2
Schrittweite      F3
Setze Breakpkt.   F4
Lösche Breakpkt   F5
Var anzeigen      F6
Var entfernen     F7
Var untersuchen   F8
PRG Bildschirm    F9
Quelltext         F10
Info anzeigen     AF1
Ausführen         AF2
Anzeige-Optionen  AF3
Fertig            AF4
Schritt überspr.  AF5
Zeit-Modul        AF6
```

*Bild 4: Der Debugger-Bildschirm mit Menü*

```
 22:     0.350---                 IF a[j] & b[i+j] & c[i-j] THEN
 23:     0.025                            x[i]:=j;
 24:     0.030                            a[j]:=FALSE;
 25:     0.030                            b[i+j]:=FALSE;
 26:     0.020                            c[i-j]:=FALSE;
 27:     0.000                            IF i<8 THEN Try(i+1) EL
                          SE print END;
 28:     0.020                            a[j]:=TRUE;
 29:     0.050                            b[i+j]:=TRUE;
 30:     0.030                            c[i-j]:=TRUE;
 31:     0.020                    END
 32:     0.355---           END
 33:     0.020            END Try;
 34:     0.000
 35:     0.000            BEGIN
 36:     0.000              FOR i:=1 TO 8 DO a[i]:=TRUE END;
 37:     0.000              FOR i:=2 TO 16 DO b[i]:=TRUE END;
 38:     0.000              FOR i:=-7 TO 7 DO c[i]:=TRUE END;
 39:     0.000              Try(1)
 40:     0.000            END AllQueens.
 41:     0.000
Hoch/Runter, ESC verlassen
```

*Bild 5: Das Zeitprofil auf dem Bildschirm*

```
27 g:\ftl » mlu wtest.lbr
MLU - FTL Modula-2 Library Utility,V1.08n
   8 Einträge vorhanden
Befehle:              F - Fertig
                      A - Addieren oder Update
                      D - Löschen
                      G - Löschen rückgängig machen
                      E - Extrahieren
                      L - Liste der Dateien
                      V - Vollständige Liste der Dateien
                      R - Reorganisieren
                      U - Update alle
                      S - Addieren scrunch Datei
                      Q - Addieren squeeze Datei
                      B - Addieren squeeze und scrunch Datei
                      Z - Alle Extender auf scrunch ändern
                      X - Alle Dateien extrahieren
                      H - diese Liste

Befehl:v
Anzahl der Einträge:   8

WTEST.PAS       2944 Sq+Scrnch  1988/12/21 13:09:48
WTEST.S         4352 Sq+Scrnch  1988/12/21 13:09:50
WTEST.BAS       3328 Sq+Scrnch  1988/12/21 13:09:54
WTEST.C         2560 Sq+Scrnch  1988/12/21 13:09:56
WTEST.MOD       2944 Sq+Scrnch  1988/12/21 13:16:34
WRSC.MOD         128            1988/07/22 19:31:56

  1 nicht verwendete Einträge.
Befehl:
```

*Bild 6: Der Bibliotheksmanager MLU*

Für andere Anwendungen läßt sich der Compiler über ein Flag zum Eincompilieren zusätzlicher TRAP-Befehle bewegen. Bindet man über den entsprechenden TRAP-Vektor einen eigenen Monitor ein, läßt sich der Programmverlauf überwachen. Ein solcher Monitor wird samt Quellcode mitgeliefert. Das Programm bleibt resident im Speicher und zeigt in der vorliegenden Form beispielsweise Prozeduraufrufe an.

## Hilfsprogramme

Wie oben schon angesprochen, gibt es für die Verwaltung der Sammelbibliotheken ein gesondertes Programm, MLU. Ähnlich wie bei den bekannten Komprimierungsprogrammen ARC oder ZOO kann es in einer Bibliothek Manipulationen wie Hinzufügen, Löschen und Extrahieren von Dateien durchführen. Neben Inhalts-Listings und Verwaltungsfunktionen wie Reorganisation des Archivs und Updates eventuell veränderter Dateien bietet eine Hilfe-Funktion eine Übersicht über die Kommandos. MLU arbeitet interaktiv per Kommandobuchstaben (Bild 6). Damit ist die Oberfläche etwas spartanisch geraten, erfüllt aber ihren Zweck.

Für größere Programmierprojekte liegt ein Make-Programm bei, mit dem bei Programmänderungen automatisch die

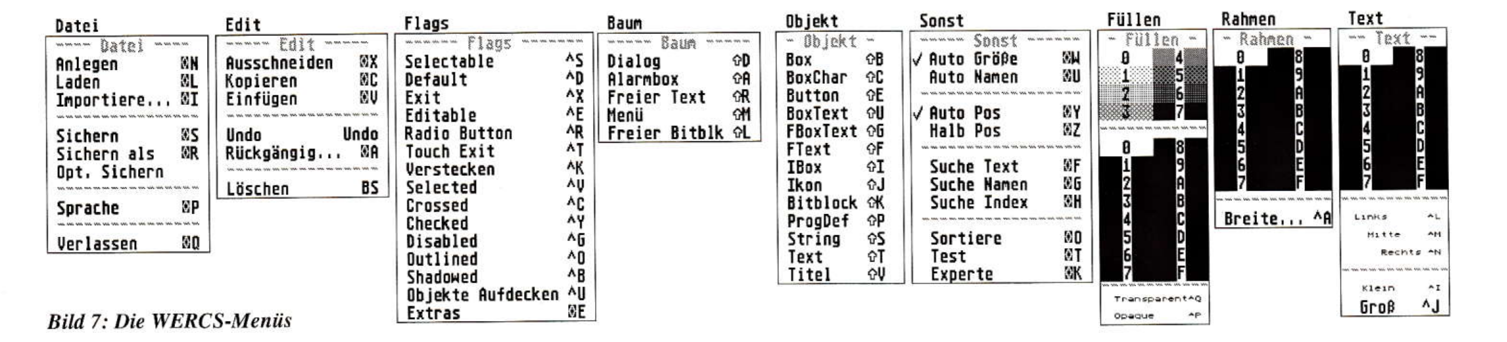



*Bild 7: Die WERCS-Menüs*

veränderten Dateien und alle von ihnen abhängigen neu compiliert werden.

Praktischerweise müssen die Abhängigkeiten zwischen Modulen nicht von Hand beschrieben werden. Dem Programm MCSCAN übergibt man einfach den Namen des Hauptmoduls, das auf Abhängigkeiten durchsucht wird, die automatisch in ein Makefile geschrieben werden.

Das Makefile kann per Hand ergänzt werden, was bei der Verwendung des Assemblers auch nötig ist. Das Make erstellt intern eine Stapeldatei für den Batcher und führt diese direkt aus, womit die Compiler-Aufrufe automatisiert sind. Bei Verwendung des Makes vom Desktop aus muß das Batchfile mit einem weiteren Hilfsprogramm gesondert erstellt werden. Die Batchfiles sind auf den mitgelieferten Batcher eingestellt und müssen für Gulām in Sachen Parameterübergabe angepaßt werden.

Für Module, die sich gegenseitig importieren, bieten die Programme PRECEDENCE und BUILDSUB eine Unterstützung beim Compilieren. Ersteres erzeugt eine Datei mit Informationen über die Abhängigkeiten zwischen den Modulen, BUILDSUB erstellt eine Batch-Datei, die den Compiler in der für die zyklischen Referenzen nötigen Reihenfolge für die Definitions- und Implementationsmodule automatisch startet. BUILDSUB scheint aber nicht richtig an das ST-System angepaßt worden zu sein: Die Batch-Datei enthält einen Aufruf "MD" - ein Programm, das nicht vorliegt. Unverständlicherweise müssen diese Programme noch aus dem vorliegenden Quellcode übersetzt werden, vielleicht sollten sie doch der Einfachheit halber lauffähig mitgeliefert werden.

Für die Erstellung von Resourcen liegt ein Resource-Construction-Set bei, und zwar WERCS, das mit FTL erstellt wurde. Das Programm entspricht in seinen Fähigkeiten den anderen bekannten Resource-Construction-Sets, ist aber an einigen Stellen nicht so komfortabel wie beispielsweise NRSC von Kuma. Zwar können Objekt mit der Maus verschoben oder verdoppelt werden, die Attribute muß man jedoch über die Drop-Down-Menüs (Bild 7) setzen.

Soll also das Füllmuster geändert werden, muß der Mauszeiger das Objekt anwählen, dann zum oberen Bildschirmrand und das Menü und den Eintrag auswählen. Eine Dialogbox, die auf Doppelklick auf das Objekt angezeigt wird, scheint doch komfortabler zu sein, wobei auch mehrere Attribute schnell zusammengesetzt werden können.

Neu für ein Resource-Construction-Set ist die Möglichkeit, die Ausgabedateien an die verwendete Programmiersprache anzupassen. Eine Textdatei legt fest, daß beispielsweise für C die Anweisung "#define" und in Modula "CONST" für Konstante verwendet werden soll. Vordefiniert sind C, Pascal, Modula, Fortran, Assembler und Basic.

## Sonstiges

Für Benutzer, die über keine Festplatte verfügen, sind zwei Utilities gedacht. Da ist zunächst ein Formatierprogramm, das die Diskettenkapazität auf 400 bzw. 800 kByte erhöht. Dazu werden die Disketten mit 10 Sektoren formatiert.

HRAMDSK ist eine resetfeste RAM-Disk, die sich per AUTO-Ordner installiert. Dabei können automatisch Dateien in die RAM-Disk kopiert werden, so daß das System dann zur Arbeit bereit ist. Zur Konfiguration der RAM-Disk gibt es ein eigenes Installationsprogramm, das in einer Dialogbox die Liste der zu ladenden Dateien, den Laufwerksbuchstaben und die Größe der RAM-Disk einstellt.

## Handbuch

Das deutschsprachige Handbuch umfaßt fast 500 Seiten und wird in einem stabilen Ringordner geliefert. Es gliedert sich in drei Hauptteile, eine Sprachreferenz über die Modula-Implementierung, ein Benutzerhandbuch, das die Entwicklungsumgebung beschreibt, und eine gesonderte Anleitung zum Resource-Construction-Set WERCS.

Die Beschreibung ist immer wieder mit Beispielen illustriert, vielleicht sollte die Ausgabe einiger Hilfsprogramme dokumentiert werden. Die genaue Gliederung erlaubt das schnelle Auffinden der gewünschten Informationen. Es existieren zwei Register für das Modula-System und das WERCS, die an einigen Stellen etwas umfangreicher sein könnten.

Ein wirkliches Ärgernis - wie schon bei MAMOS-Modula - ist das Fehlen von Listings der Definitionsmodule. Zwar beschreiben cirka 100 Seiten des Handbuchs die Module, insbesondere bei den Standardmodulen findet man aber keine Angaben über die Parameterstrukturen. Auch sind exportierte Datentypen nicht dargestellt. Zum schnellen Nachschlagen ist dieser Teil nicht geeignet.

Es bleibt also nur das langwierige Ausdrucken aller Definitionsmodule. Heraus kommen über 100 Druckerseiten, die natürlich nicht in das DIN A5-Handbuch eingeheftet werden können und einen unordentlichen und unhandlichen Stapel bilden. Das mitgelieferte LIST-Programm führt übrigens keine vernünftige Paginierung durch, so daß das Ausdrukken nicht sonderlich unterstützt wird.

Das Argument, daß die Bibliotheken ständig erweitert werden, kann nicht gelten, immerhin liegt das Handbuch auch hier als Ringbuch vor. Ergänzungen sind so kein Problem; ein Verfahren, das bei SPC-Modula hervorragend funktioniert.

Das Handbuch ist also annehmbar und wird eigentlich alle Fragen beantworten. Ohne eine Ergänzung um die Listings der Definitionsmodule ist es aber für die tägliche Arbeit unvollständig.

## Fazit

Der erste Eindruck des Pakets mit den vielen Hilfsprogrammen, der angebotenen Zeitprofil-Analyse und der kompakten Speicherung der externen Module ist positiv. Einige Konzepte weisen in die richtige Richtung, wären sie doch nur fehlerfrei implementiert. Bei genauer Betrachtung zeigt sich eine Reihe Mängel, die die Qualität des Pakets letztlich beeinträchtigen.

FTL-Modula deutet eine sinnvolle Ausstattung an, ist aber an vielen Stellen noch lange nicht ausgereift. Eine neue Version könnte interessant werden; hoffentlich nutzen die Entwickler die Zeit, denn momentan kann FTL noch nicht gegen die breite Modula-Konkurrenz bestehen.

*RT*

***Bezugsadresse:***

*CCD*
*Creative Computer Design*
*Burgstr. 9*
*6228 Eltville*
*Tel.: 06123/1638*

# Fort-Schrittlich

## Turbo C 2.0 und sein Source-Level-Debugger

**Als Borland vor einigen Monaten auf dem ST-Markt in Erscheinung trat, waren viele überrascht. Nicht Turbo Pascal, das immer wieder angekündigt worden war und bis heute nicht erschienen ist, war nun zu erwerben, sondern Turbo C wurde Borlands erstes Zugpferd. Die Kunden stürzten sich geradezu auf den neuen mit ANSI-Standard ausgerüsteten C-Compiler. War es bisher der Megamax-Compiler, der den Markt beherrschte, so gab es nun auf dem Markt zwei Compiler, die konkurrierten. Während es von Megamax schon länger die neue Laser-C-Version gibt, kam Mitte letzten Jahres der erste Source-Level-Debugger für den ST auf den Markt, der Laser-C stark aufwertete. Jetzt schlägt Borland mit Turbo 2.0 zurück, das als Unterstützung ebenfalls einen Source-Level-Debugger zur Seite bekommt.**

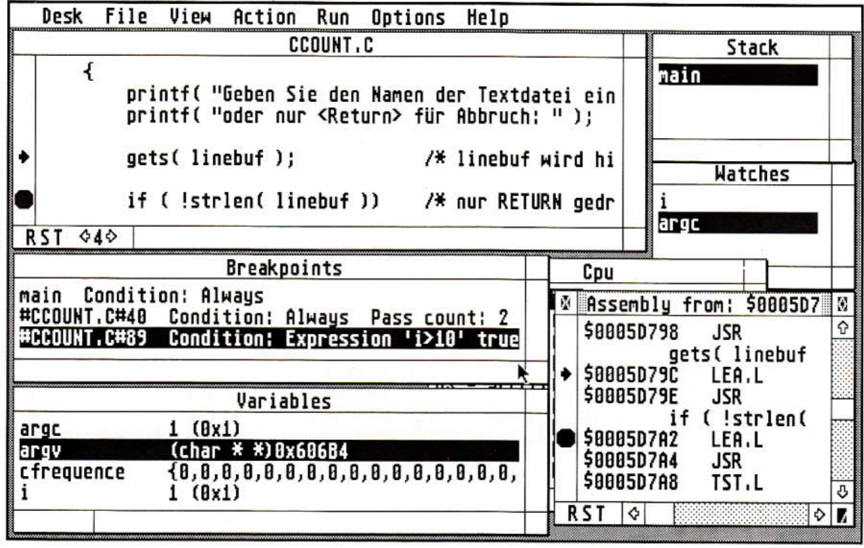



*Bild 1: Durchblick mit vielen Fenstern*

Borland hat es schon schwer. Nicht nur, daß sie sich mit der Konkurrenz auf dem ATARI messen müssen, auch die Software auf dem IBM (und dessen kompatiblen) ist ein Vorbild. Dies ist sicherlich eine schwierige Aufgabe. Ich vermute nämlich, daß Borland immer die Entwicklung auf dem IBM-Markt forcieren wird. Schade sicherlich für Turbo Pascal-Anhänger, die gerne auch Turbo Pascal auf dem ATARI sehen würden. Aber dies ist ein anderes Thema und vielleicht auch noch nicht abgeschlossen - Borland ist immer für Überraschungen gut. Beschäftigen wir uns also etwas eingehender mit dem neuen Turbo C.

### Das neue Turbo C 2.0

Wenn man ehrlich ist, muß man zugeben, daß sich in Sachen Turbo C relativ wenig im Vergleich zum Turbo Debugger getan hat. So finde ich es schon fast etwas übertrieben, dieser Version eine um eins erhöhte Nummer zu geben, da dadurch normalerweise umfangreiche Änderungen angekündigt werden. Trotzdem sind auch diese hier erwähnenswert. Zunächst möchte ich den neuen Linker erwähnen, der in seiner neuen Gestalt nicht nur das allseits bekannte Digital Research-Format, sondern auch ein neues, von Borland definiertes Format linkt, welches von Turbo C erzeugt wird. Dieses Format ist so kompakt, daß sich ein separates Bibliotheksformat nicht lohnt. Das Link-Format ersetzt damit die übliche Bibliotheksstruktur, da der Linker sowieso nur die benötigten Dateien zum Hauptprogramm hinzufügt. Des weiteren fordert der ANSI-Standard die Mindestlänge eines Bezeichners von 32 Zeichen, die mit nur acht Zeichen bei bei Digital Researchs Link-Format nicht gegeben ist - Borland hat es sogar auf 255 Zeichen erweitert. Ab Turbo C 2.0 kann eine Kommando-Shell angegeben werden, die, aus der C-Shell heraus, ausführbar ist.

### Vergeblich

Wer darauf gewartet hat, daß Turbo C 2.0 endlich einen Inline-Assembler enthält, hat vergeblich gewartet. Zugegeben, es gibt einen Makro-Assembler im Professional-Paket, aber für mich ist der nicht eingebaute Inline-Assembler immer noch das größte Manko von Turbo C überhaupt (Portabilität kann man, wenn man unbedingt will, durch *conditional defines* erreichen). Daß diese Test-Autoren auch immer etwas zu meckern haben... Wahrscheinlich ist das o.g. Politik, denn auf MS-DOS-Rechnern gibt es sowas wie einen Inline-Assembler in Turbo C auch nicht!

### Standardisierte Grafik

Als wohl umfangreichste Erweiterung des Turbo C kann man sicherlich die Grafikbibliothek nennen. Schon vor einigen Jahren von Borlands IBM-Programmierern aus der Taufe gehoben, stellt sich nun auch auf dem ST der BGI-Treiber (Borlands Graphic Interface) vor. Mit diesem Treiber ist es standardisiert möglich, Grafiken (Linien, Kreise, Füllmu-

ster, Buchstaben usw.) auf dem Bildschirm auszugeben, wobei der Treiber für die gängigen Grafikkarten ausgelegt wird. Sicherlich könnte man einwenden, daß man dafür auf dem ATARI das VDI hat, was sicherlich richtig ist. Nur können Programme, die den BGI-Treiber nutzen, direkt auf den IBM portiert werden, wobei dies natürlich auch für den umgekehrten Fall (IBM -> ST) gilt. Auch die dafür definierten Vektorzeichensätze lassen sich vom ST auf den IBM (und umgekehrt) portieren. Den Umfang der Anwendung kann ich noch nicht abschätzen, es werden aber sicherlich einige Entwickler auf dem ST diese Tatsache begrüßen. Ein Vektor-Font-Editor wird nicht mitgeliefert und kann (soweit mir bekannt) auch nicht erworben werden, es sind aber zehn Zeichensätze vorhanden.

## Flags

Natürlich ist Turbo C 2.0 ab sofort in der Lage, Debug-Informationen in den Code zu integrieren. Dazu gibt es die Y-Option beim Compiler und beim Linker. In der Shell ist (schon seit der Version 1.1) ein Menüpunkt vorhanden, der nach dem (eventuell nötigen) Compilieren automatisch den Debugger aufruft. Wer jetzt denkt, daß dabei automatisch das Y-Flag für Compiler und Linker gesetzt werden, hat weit gefehlt. Dadurch ist es mir nicht nur einmal passiert, daß ich mich nach geglücktem Compiler-Lauf im Debugger befand, der natürlich nichts besseres zu tun hatte, als eine Meldung auszugeben, der Code habe keine Debug-Information; selbst ein Compiler-Lauf, der mit Debug eingeleitet wurde, führt bei einem Link-Error trotzdem zum Aufruf des Debuggers...!?!? Ärgerlich ist auch, daß, wenn Flag-Einstellungen innerhalb der Shell geändert wurden (Beispiel: Y-Flag in Compiler und Linker eingeschaltet), die Make-Option dies nicht erkennt. Mit anderen Worten: Ist das Programm vorher schon einmal ohne Y-Einstellungen compiliert worden, weigert sich die Shell standhaft, mit MAKE neu zu compilieren. Sicherlich gibt es Fälle, bei denen man Debuggen will und Teile nicht mit Debug-Info versehen möchte - ich möchte aber behaupten, das sei der seltenere Fall. Ich hoffe, daß das automatische Setzen des Y-Flags bei DEBUG und das Erkennen von Änderungen von Optionen durch MAKE noch implementiert wird, denn die Shell soll ja Arbeit abnehmen! Man bedenke, daß man sonst eigentlich nicht viel an ihr aussetzen kann, denn fast alles läßt sich mit der Tastatur steuern (lobenswert!). Wenn da nicht der Editor wäre: Mir ist bewußt, daß ich ab diesem Moment alle GEM-Fans gegen mich habe, aber warum muß dieser Editor so langsam sein? Ich weiß, daß er ausschließlich mit GEM-Routinen programmiert ist und daher auf jeder Grafik laufen wird. Dies ist aber nur eine Vereinfachung der Programmierung von Borlands Seite. Ich als Anwender habe mit einem solchen Editor zu kämpfen. Ich gebe zu, daß man die unsaubere Programmierung übertreiben kann, aber trotzdem bin ich ein Anhänger kurzer Turn-Around-Zeiten, die mit diesem Editor auf gewisse Weise zunichte gemacht werden. Daß es auch anders geht, zeigen die Windows im Turbo-Debugger, die angenehm zu bedienen sind.

## Turbo-Debugger

Womit wir beim Thema wären: Der Debugger ist in der Lage, alle Programme, die mit Turbo C geschrieben wurden, zu debuggen. Das heißt, daß er auch GEM-Programme debuggen kann, womit indirekt erwähnt ist, daß er das Original-GEM des ATARIs nicht verwenden darf. Deshalb wurden die entsprechenden Routinen nachprogrammiert und so müssen Sie auch im Debugger nicht auf Drop-Down-Menüs und Fenster verzichten. Im Gegenteil, alles funktioniert merklich schneller und trotzdem ist er lauffähig auf Großbildschirmen (dafür mag er die Farbgrafik überhaupt nicht). Wo bleibt die Konsequenz? Turbo Cs Editor wird wegen Grafik-Kompatibilität nicht schneller - und beim Debugger macht man es doch. Hoffen wir für Borland, daß es in Zukunft nicht allzu viele Grafikkarten unterschiedlicher Machart für den ST gibt: Ich bin jedenfalls begeistert von der Bedienerfreundlichkeit und Arbeitsweise des Turbo Debuggers. Daß man es hier nicht mit GEM zu tun hat, merkt man spätestens an der Tatsache, daß sich eine Menge Fenster öffnen lassen (beispielsweise sieben Fenster gleichzeitig, wie in Bild 1 zu sehen), auch wenn es die Übersichtlichkeit nicht unbedingt fördert. Dennoch hat man dadurch die Möglichkeit, seine Debugging-Informationen so darzustellen, wie man möchte.

## Source-Level-Debugging

Bevor ich darauf eingehe, was Borlands jüngstes Kind so alles kann, möchte ich für alle, die noch nicht wissen, was ein Source-Level-Debugger ist, den Begriff kurz erläutern. Fehler in einem Programm hat sicherlich schon jeder gesucht. Die gängigste Methode dafür in Hochsprachen ist beispielsweise das Einfügen von Tastaturabfragen und Ausgaben von bestimmten Informationstexten oder auch Variablenwerten. Daß dies keine schöne Methode und auch alles andere als schnell ist, kann sicher jeder bestätigen, der schon so vorgegangen ist. Assembler-Programmierer kennen schon viel länger die Möglichkeiten, ein Programm schrittweise, (Single-Step) oder Teile eines Programmes bis zu einem Breakpoint (Haltemarkierung) abzuarbeiten. Dabei bietet der Debugger die Möglichkeit, Register oder Speicherbereiche darzustellen und diese auch zu ändern. Ein solches Debugger-

```
File                  View                 Action                Run
Load...          ^O   Function...     ^S   Find...          ^F   Run              ⌧R
Info...          ^I   Breakpoints     ^B   Find Same        ^G   Trace into       ⌧T
                      Variables       ^V   Goto Line...     ^L   Step over        ⌧S
Close Window     ^U   Watches         ^E                         Until return     ⌧U
Cycle Windows    ^W   Modules         ^M   Breakpoint...    ⌧B   Animate          ⌧A
                      Stack           ^T   Inspect...       ⌧I
Quit             ^Q                        Watch...         ⌧W   Program reset    ⌧Z
                      Assembly...     ^A   Log...           ⌧L
                      Dump...         ^D
                      CPU             ^C   Change...        ⌧C
                                           Edit...          ⌧E
                      File...         ^H   Block...         ⌧K
                      Log             ^J   Delete           DEL
                      User Screen     ESC  Delete all

Options                      Help
Environment          ⌧N      Menu
Directories          ⌧D      Debugger
Arguments            ⌧P      C Language
                             Libraries
Load...              ⌧J      Assembler
Save "TD.CFG"        ⌧H      Index
```

*Bild 2: Die reichhaltigen Funktionen des Debuggers*

Konzept für Hochsprache zu realisieren, ist natürlich nur mit einem vergleichsweise hohen Aufwand möglich. Trotzdem gibt es inzwischen auf vielen Rechnern solche Debugger, die es dem Programmierer ermöglichen, auf Quelltextebene (source-level) ein Programm in Einzelschritten oder bis zu einem Breakpoint zu bearbeiten und dabei Variablen zu untersuchen. Wichtig ist, daß der Debugger 'standhafter' ist als das eigentliche Programm, das heißt, daß der Debugger nicht unbedingt in Mitleidenschaft gezogen wird, wenn das zu debuggende Programm nicht das tut, was es eigentlich soll. Soviel zum Konzept, nun zur Praxis von Borlands Realisierung.

## Schlaraffenland

Jemand, der noch nie mit einem Source-Level-Debugger gearbeitet hat, fühlt sich am Anfang wahrscheinlich wie erschlagen von der Funktionsvielfalt, die der Turbo Debugger zu bieten hat (siehe Bild 2). Dennoch wird es ihm relativ leicht gemacht, da erstens die von Turbo C bekannte Online-Hilfe auch für den Debugger erhältlich ist, und zweitens das Handbuch kaum einen Wunsch offen läßt. Das Handbuch zeichnet sich durch 160 Seiten, ein umfangreiches Inhaltsverzeichnis und einen ausreichendem Index aus. Gut ist, daß nicht nur das Programm selbst dokumentiert ist, sondern in einem Kapitel der Ablauf eines Debug-Laufs an einem Beispiel erklärt wird. Außerdem werden prinzipielle Vorgehensweisen beim Debuggen aufgezeigt. Wahrscheinlich ist das Handbuch wieder so umfangreich, daß viele es nicht lesen werden (Wie sagte schon mein Kollege J. Leonhard: "Real programmers don't read handbooks") - wie man es macht, ist es falsch. Das Handbuch verdient sicherlich ein Lob.

## Untersuchung

Prinzipiell kan man behaupten, es ist alles da, was man so zum Debuggen braucht. Viele der Dinge, die ich im folgenden erkläre, können Sie in Bild 1 näher betrachten. Der Quelltext oder Assemblercode kann im Einzelschritt (Pfeil zeigt auf aktuellen Befehl) durchgegangen werden oder bis zu einem bestimmten Breakpoint (dicker Punkt im linken Teil des Quelltext- oder Assembler-Fensters). Dabei ist besonders erwähnenswert, daß man gleichzeitig in zwei verschiedenen Fenstern Assembler und Quelltext bearbeiten kann, obwohl sogar im Assembler-Fenster die Quelltextzeilen mit in den Assembler-Text 'eingestreut' werden. Fast selbstverständlich erscheint da schon die Tatsache, daß man alle Breakpoints übersichtlich auflisten und sie dabei auch abhängig von bestimmten Bedingungen setzen kann. Dadurch muß also nicht immer an einer bestimmten Stelle angehalten werden, sondern beispielsweise nur beim 10ten Mal oder dann, wenn sich eine bestimmte Speicherstelle geändert hat. Wichtig für den Anwender ist, daß sich dadurch der Programmablauf, auch wenn man sich nicht im Single-Step befindet, verlangsamt, da ja der Debugger vor jedem Befehl alle Bedingungen überprüfen muß, die zum Abbruch führen könnten. Damit Sie nicht alles im langsamen Single-Step oder im schnellen Ablauf bis zu Breakpoints testen müssen, gibt es noch ein Mittelding, die sogenannte Animation. Bei dieser Animation wird ein Single-Step ausgeführt, der (einstellbar) zeitgesteuert das Programm selbständig durchläuft.

Mit einer der wichtigsten Eigenschaften eines Source-Level-Debuggers ist das Anzeigen von Variablen: Dies kann global, statisch, lokal oder funktionsbezogen durchgeführt werden. Beispielsweise können Sie sich aller Variablen darstellen lassen, die momentan lokal von einer bestimmten Unterroutine verwendet werden. Dabei werden die Variablen in gängiger C-Systax dargestellt, so daß auch Felder im Fenster erscheinen. Zeiger auf Strukturen werden zunächst als Adressen dargestellt, können aber durch INSPECT und ein weiteres kleines Fenster (on-line) in den einzelnen (bezeichneten) Elementen untersucht werden. Nur der Vollständigkeit halber erwähne ich noch zum Schluß, daß es möglich ist, Speicherbereiche anzuschauen, Dateien über einen startbaren Editor (nicht im Turbo Debugger selbst) zu ändern und Register der CPU anzuschauen. Sie können ein Log-File erstellen, in dem alles festgehalten wird, was während des User-Programmablaufes passiert (Breakpoints, Absturz Ihres Programms etc.). Diese Ausgaben eines Anwenderprogramms können natürlich nachwievor ausgeführt werden, da diese auf einen anderen Bildschirm umgeleitet werden. Der Debugger kann so eingestellt werden, daß bei einer Ausgabe, die er erkennen kann, automatisch auf den anderen Bildschirm umschaltet. Praktisch ist, daß man alle Einstellungen (wie bespielsweise die Fenster-Konfiguration) abspeichern kann.

## Zusammenfassung

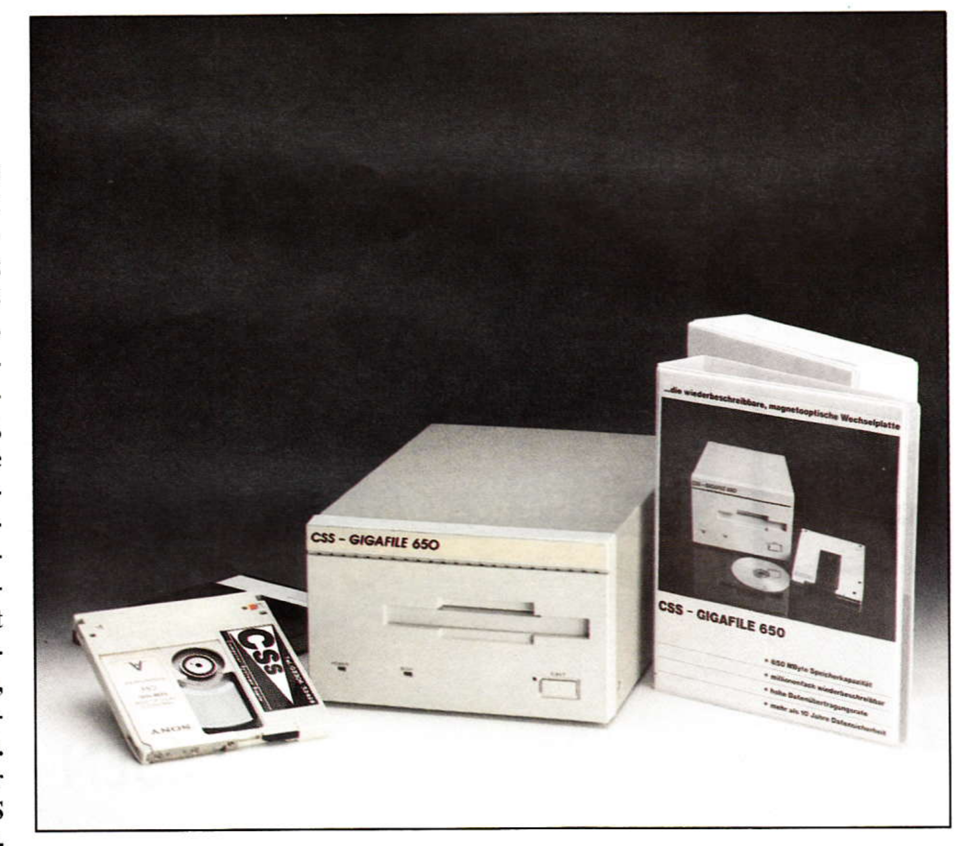
Sicherlich hat Borland mit dem Debugger ein neues, hervorragendes Produkt auf den Markt gebracht - der Debugger scheint gut debuggt zu sein... Trotzdem fragt man sich, warum nicht, wie bei Turbo C auf dem IBM, ein in seinen Funktionen nicht so umfangreicher Source-Level-Debugger in die Turbo C-Shell integeriert wird. Nicht immer braucht man die umfangreichen Möglichkeiten des Turbo Debuggers, den es in der MS-DOS-Version zusätzlich zum eingebauten Debugger zu kaufen gibt. Es wäre sicherlich wünschenswert gewesen, wenn die Shell etwas mehr überarbeitet worden und endlich ein Inline-Assembler in Turbo C vorhanden wäre. Trotzdem erhält man sicherlich inzwischen ein absolut professionelles Paket, indem man Turbo C 2.0 und Debugger kauft. Allerdings ist ab sofort das Preisverhältnis ein anderes zum Hauptgegner Laser-C. Turbo C 2.0 für den ATARI ST wird ab März 1990 erhältlich sein und kostet 248.52 DM. Das Professional-Paket enthält Turbo C 2.0, den Assembler MASM und den neuen Turbo Debugger und kostet 458.28 DM. Hält man dagegen den Laser C-Compiler, der ab März 349,- DM. inklusive Source-Level-Debugger kostet, wird es sicherlich keine einfache Entscheidung für den Kunden, der noch keinen C-Compiler besitzt.

*SH*

*Bezugsadresse:*

*Heimsoeth + Borland*
*Lindwurmstr. 88*
*8000 München 2*
*Tel.: 089/720100*

# Optischer Riese




**Mal mußte es so kommen: Kaum haben wir uns - ehemals noch stolz auf glatte 720 kB Diskettenkapazität - an die Megabytes auf den Festplatten gewöhnt, wollen uns alerte Nachswuchskonzerne mit gigantischen Datenschluckern beglükken. Genauer: Mit einem optomagnetischen Laufwerk, das 650 Meckerbytes beherbergt. Es kommt von der jungen Firma CSS und nennt sich GIGAFILE 650.**

Seit einigen Jahren gibt es bereits optische Laufwerke, die allerdings nicht (CD-ROM) oder nur einmal beschreibbar (WORM-Laufwerke) sind. Jetzt endlich sind die löschbaren optomagnetischen Laufwerke zur Serienreihe gediehen, Sony ging mit den ersten durchs Ziel. Das optomagnetische Laufwerk Sony SMOD 501 hat die Firma CSS als eine der ersten an den ST angepaßt.

## Neue Medien

Das Medium erinnert stark an eine CD; in einer Polykarbonatschicht ist eine reflektierende Aluminiumschicht eingebettet. Auf die Aluminiumschicht ist ein kristallines, magnetooptisches Substrat aufgebracht. Ein Laser und ein Magnetkopf vollbringen zusammen das Wunder der Geburt von Informationen auf dieser Schicht: Beim Schreiben wird das Magnetfeld im Schreibkopf zuerst in "0"-Richtung gepolt. Der Laser erhitzt das magnetooptische Substrat bis auf eine Temperatur, die man *Curie-Punkt* nennt: Bei dieser Temperatur "vergessen" die Kristalle im Substrat ihre Orientierung und richten sich nach dem anliegenden äußeren Magnetfeld aus. Bei diesem Durchgang wird also zuerst eine komplette Spur oder ein Sektor gelöscht. Danach wechselt das Feld im Magnetkopf in die "1"-Richtung. Immer wenn ein Bit unter dem Magnetkopf vorbeikommt, das auf 1 gesetzt werden soll, erhitzt der Laser dieses "Bit" bis zum Curie-Punkt, woraufhin es die magnetische "1"-Orientierung annimmt.

Warum löscht und schreibt man nicht in einem Durchgang? Das Problem: Zur Erzeugung des Magnetfelds sind recht große Spulen nötig, deren Feld man nicht schnell genug umpolen kann, um mit den am Kopf vorbeihuschenden Bits Schritt zu halten. Deswegen löscht man zuerst alle Bits, polt dann um und schreibt einzelne "1"-Bits. Daraus folgt, daß das Schreiben auf magnetooptischen Laufwerken doppelt so lange dauert wie das Lesen.

Beim Lesen wird der Laser mit wesentlich kleinerer Intensität auf die Aluminiumschicht gerichtet; er wird von dort reflektiert und durchquert dabei auch das magnetooptische Substrat. Dieses Material hat aber eine besondere Eigenschaft: Je nach der Orientierung der Kristalle wird das einfallende Licht verschieden *polarisiert*, das heißt, die Schwingungsrichtung wird unterschiedlich gedreht (*magnetooptischer Kerr-Effekt*, Bild 1). Diese Polarisierung im reflektierten Licht kann man messen und in binäre Information wandeln.

Anders als bei magnetischen Medien ist der Informationsträger in einer Schutzschicht geborgen. Zudem ist der Abstand der Leseeinrichtung bei optischen Laufwerken wesentlich größer als bei Festplatten - das Wort "Headcrash" kann man gleich wieder aus dem Gedächtnis streichen. Dazu kommt, daß bei optischen Medien raffinierte Verfahren zur Fehlererkennung und -korrektur angewendet

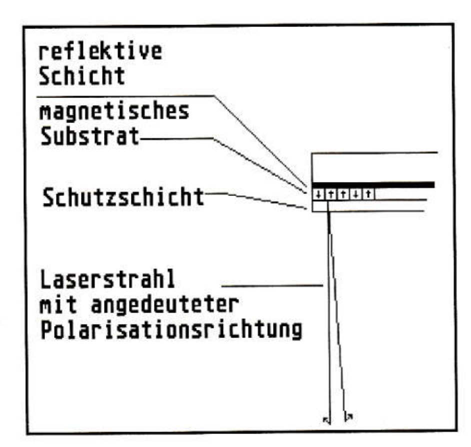



*Bild 1: Der magnetooptische Kerr-Effekt*




*Bild 2: Neue Medien braucht das Land.*




*Bild 3: Die Rückseite der GIGAFILE 650*

werden. Kein Wunder, daß Sony eine Bit-Fehlerwahrscheinlichkeit von $10^{-12}$ garantiert - nur ein Bit von $10^{12}$ wird also im Durchschnitt falsch gelesen. Das heißt: Statistisch gesehen ist auf einem von 1830 Medien ein einziges Bit defekt. Sony geht sogar soweit, auf das Medium 10 Jahre Datengarantie zu geben!

Weil das Medium so unempfindlich ist, kann man auch gegen von außen eindringenden Staub weit toleranter sein als bei magnetischen Medien (Festplatten): Das Medium wird wechselbar - mit allen positiven Konsequenzen.

## Theorie und Praxis

Das Medium von Sony (Bild 2) nennt sich kryptisch EDM-1DA1 und ist in einer schützenden stabilen Plastikkassette verborgen - ähnlich wie bei den 3.5"-Disketten, nur daß die Plastikhülle mehr dem 5.25"-Format entspricht. Auch hier findet man Schreibschutzschieber (für jede Seite getrennt) und einen Schutzschieber, der das eigentliche Medium solange verbirgt, bis die Plastikkassette in das Laufwerk gesteckt wird.

Das Medium ist zweiseitig und hartsektoriert. Das bedeutet, daß man es nicht mehr formatieren muß - die Einteilung in 512-Byte-Sektoren ist bereits im Werk geschehen. Es gibt auch eine Variante des Mediums, bei der die Sektoren 1024 Bytes groß sind. Dadurch kann mehr Platz auf der Spur ausgenutzt werden, Kapazität und Datendurchsatz steigen. Leider gibt es bisher keinen Plattentreiber für den ST, der mit größeren *physikalischen* Sektoren klarkommt, weswegen CSS ihr Laufwerk mit dem 512-Byte-Medium ausliefern. Damit kommt man immerhin noch auf eine Kapazität von 2*297 MB (=594 MB, beim 1024-Byte-Medium sind es 650 MB).

Das physikalische Format des Mediums entspricht dem ISO-Standard Typ A für optomagnetische Laufwerke, was im noch heiß umkämpften Markt für die neuen Speicherriesen bedeutet, daß man - hoffentlich - auch später noch leicht an Medien für dieses Laufwerk kommt. Zudem ist Sony im Moment Marktführer auf diesem Gebiet.

## Das Laufwerk

Das Laufwerk Sony SMO-D501 verfügt nur über eine Schreib-/Lesevorrichtung, weswegen immer nur auf eine Seite des Mediums zugegriffen werden kann. Um die Rückseite ebenfalls lesen und beschreiben zu können, muß man es umdrehen. Zwar halbiert das die jederzeit erreichbare Datenmenge auf knapp 300 MB, es spart aber auch Kosten.

Nach dem Einschieben wird das Medium auf 2400 Umdrehungen in der Minute beschleunigt; das dauert knapp 6 Sekunden. Beim Entnehmen fährt das Laufwerk in weniger als drei Sekunden herunter; das sind, im Vergleich zu einer Syquest-Wechselplatte (MEGAFILE 44, profile 44 etc.), recht angenehme Zeiten.

Am Laufwerk, das einen ESDI-Bus anbietet (ein bei PCs verbreiteter Peripheriebus), wird ein Controller angeschlossen, der den ESDI-Bus in den SCSI-Standard wandelt. Damit genießt man bei der CSS GIGAFILE alle Annehmlichkeiten, die man von SCSI-Geräten gewohnt ist -

allerdings werden die zum größten Teil durch den nachgeschalteten Host-Adapter, der den SCSI-Bus an ATARIs ACSI-Bus anpaßt, zunichtegemacht. Woran nicht der Host-Adapter schuld ist, sondern das etwas engstirnige Design des ACSI-Busses. Immerhin profitiert man noch vom 64 kB großen Cache-Speicher auf dem SCSI-Controller; ein weiterer Vorteil des SCSI-Controllers ist, daß ein und dasselbe Laufwerk sehr leicht auch am Mac, an PCs, am AMIGA und am SCSI-Anschluß des TT eingesetzt werden kann - passende Software vorausgesetzt.

## Von Turnschuhen und Früchten

Das Laufwerk kommt in voller 5.25"-Bauhöhe daher - heutzutage ein bemerkenswerter Anblick, hat man sich doch so sehr an Plättchen gewöhnt, die allenfalls ein Zoll hoch sind. (Der Stand der Flunder-Technik Ende 1989 war 0.6 Zoll!) Die CSS GIGAFILE 650 ist also etwa doppelt so hoch wie ein MegaST-Gehäuse, aber nur so breit wie eine SH204 - aber ansehnlicher und flotter anzuschauen als weiland die SH204, also sagen wir: Sportschuhkarton-Format.

Die Front ist hellgrau gegen das restliche Gehäuse (ATARI-grau) abgesetzt; dort findet man eine Aktivitätsleuchte und eine Einschalt-LED, eine Öffnung für das Medium sowie die Auswurftaste des Laufwerks. Diese Auswurftaste kann via Soft- oder Hardware blockiert werden, um das Auswechseln oder Entwenden des Mediums zu verhindern. Allerdings kann man diesen Schutz überlisten, indem man mit einem spitzen Gegenstand in einem Loch neben der Auswurftaste den "Emergency Exit" betätigt. Erinnert mich irgendwie an Tomaten - oder waren's doch Bananen?

## Kontakte zur Außenwelt

Das Heck des sportiven Schuhkartons (Bild 3) zieren ein Netzschalter, ein Mäuseklavier (vulgo: DIPs), zwei 50polige SCSI-Bus-Anschlüsse und ein 5V-Anschluß für den ACSI-Host-Adapter. Sie folgern richtig, der Host-Adapter ist ein separates Accessoir, an zwei Schnüren (1*Saft, 1*SCSI) an der Schulter zu tragen. CSS liefert zur Zeit einen Host-Adapter der Firma GESOFT aus. Dessen Kompatibilitätsprobleme mit dem Supercharger sind inzwischen ausgestanden, das ganze Gespann kann man somit ohne Zögern als "AHDI-kompatibel" klassifizieren (siehe [1]), zumal AHDI und - nach einem kleinen Patch - auch HDX damit laufen. Der Host-Adapter bietet zwei DMA-Anschlüsse, darunter einen gepufferten DMA-Ausgang; die Target-Adresse des Host-Adapters stellt man nicht direkt an diesem ein, sondern am SCSI-Laufwerk, in diesem Falle auf dem Mäuseklavier auf der Rückseite des Hauptgehäuses.

Der Grund für die Zweiteilung von Host-Adapter- und Hauptgehäuse ist, daß im Laufwerksgehäuse wenig Platz bleibt (Bild 4): Das Schaltnetzteil, das sich automatisch auf alle Wechselspannungen von 110 bis 240 V und auf 50 oder 60 Hz Netzfrequenz einstellt, der 12V-Lüfter, das Laufwerk und der SCSI-Controller sind zu raumgefräßig.

| | GIGAFILE | GIGAFILE (Angaben lt. Sony) | SH205 ST225-Laufwerk) | LACOM SD40Q (Quantum-Laufwerk) |
|---|---|---|---|---|
| minimale Spurwechselzeit | 6 ms | 10 ms | 13 ms | 5 ms |
| mittl. Spurwechselzeit | 98 ms | 95 ms | 73 ms | 24 ms |
| max. Spurwechselzeit | 184 ms | 185 ms | 150 ms | 36 ms |
| Transferrate m. Zylinderwechsel | 206 kB/s | k.A. | 408 kB/s | 778 kB/s |
| ...ohne Zylinderwechsel | 310 kB/s | 620 kB/s | 510 kB/s | 778 kB/s |

*Tabelle 1: Hardware-Kenndaten der GIGAFILE im Vergleich*

| | GIGAFILE | MEGAFILE 60 | protar 40DC | 9-Sektor-Disk |
|---|---|---|---|---|
| **TRANSFER.PRG** | | | | |
| Transfer an gerade Adressen | 333 kB/s | 545 kB/s | 700 kB/s | 17.2 kB/s |
| **HDBENCH** | | | | |
| Dateien anlegen | 27.7 s | 11.6 s | 10.5 s | 180.0 s |
| Dateien lesen | 2.4 s | 1.7 s | 1.3 s | 28.5 s |
| Dateien löschen | 11.0 s | 3.2 s | 2.9 s | 62.0 s |

*Tabelle 2: Benchmarks in der Oberschicht*

## When the wind blows...

Ach ja - zu meinem Lieblingsthema "Lüfter und ihre sozioökologischen Auswirkungen", man kennt das: Dieser hier ist zwar kein verwunschener Staubsauger wie sein Verwandter in der MEGAFILE 44, aber deutlich hörbar. Auch das Laufwerk entwickelt einen beachtlichen Geräuschpegel. Immerhin schien mir die GIGAFILE angenehmer als meine wakkere SH205 (aufgerüstet auf 40 MB) und deutlich leiser als eine MEGAFILE 44 - keine Kunst, Ihrem Einspruch sei stattgegeben. Immerhin rauscht die GIGAFILE 650 in einem für mich weniger anstößigen Frequenzbereich, und da der Host-Adapter vom Laufwerk getrennt ist, kann man letzteres zum Glück relativ weit entfernt vom Rechner aufstellen - zur Länge des DMA-Kabels (sollte keinesfalls länger als 1.2 m sein, im Normalfall sind es zwischen 50 und 90 cm) addiert sich die Länge des SCSI-Kabels vom Host-Adapter zum Laufwerk.

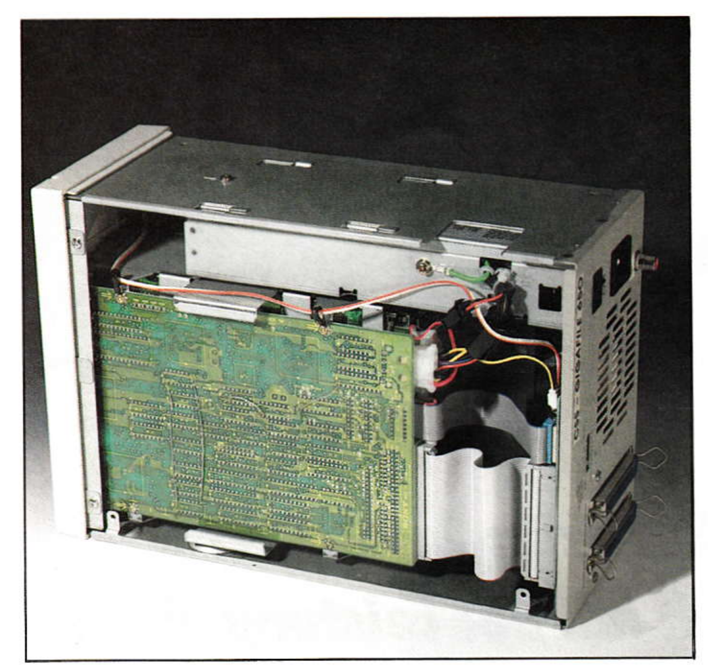


*Bild 4: Trautes Innenleben*

## Lustige Laufwerks-Hatz

Optomagnetische Laufwerke erheben den Anspruch, irgendwann einmal die reinen magnetischen Laufwerke abzulösen. Ob das gelingt, hängt außer vom Preis/Leistungsverhältnis und dem Beharrungsvermögen des Marktes zu einem großen Teil von der Geschwindigkeit der Laufwerke ab. Benchmarks gehören also auch hier zum täglich' Brod eines jeden Testers...

Man betrachte hierzu Tabelle 1. Dort sind Werte aufgeführt, wie sie von CHECK-HD, dem hardwarenahen Platten-Benchmarkaus dem Benchmark-Paket der ST-Computer, für die GIGAFILE 650 und für verschiedene Festplattentypen ausgespuckt werden.

Daß maximale und mittlere Spurwechselzeiten nicht so kurz sind wie bei typischen Festplatten, ist kein Wunder: Pro Seite verfügt so ein Medium über 18751 Spuren, und die Meßwerte hängen davon natürlich ab. Fairerweise muß man bedenken, daß man während einer Arbeitssitzung meist nicht auf die vollen 300 MB einer Seite zugreift, sondern nur auf einen kleinen Ausschnitt davon, oft nur auf eine Partition. Für einen solchen eingeschränkten Arbeitsbereich sind die Spurwechselzeiten etwas geringer anzusetzen.

Unangenehmer ist da schon die relativ niedrige Transferrate. Sony gibt eine Rate von 620 kB/s an, was auch vernünftig klingt: Bei 15672 Bytes pro Spur (31 Sektoren zu je 512 Bytes) und 40 Umdrehungen in der Sekunde (2400 UpM) kommt man genau auf diesen Wert. Tatsächlich messen wir jedoch exakt die Hälfte, also 310 kB/s. Über die Diskrepanz habe ich mich doch gewundert; ich konnte mir das nur so erklären, daß das Laufwerk (bzw. der SCSI-Controller) aus irgendeinem Grund mit Interleave 2 arbeitet, also zwei Umdrehungen braucht, um eine Spur komplett einzulesen.

Man sieht: Die Geschwindigkeit der Festplatten erreicht das Sony-Laufwerk noch nicht, als Leib- und Magenlaufwerk eines heftigen Programmierers taugt es nicht ganz. Für die Mehrzahl der Anwender, die nicht ganz so durchsatzverliebt sind, reicht es aber aus.

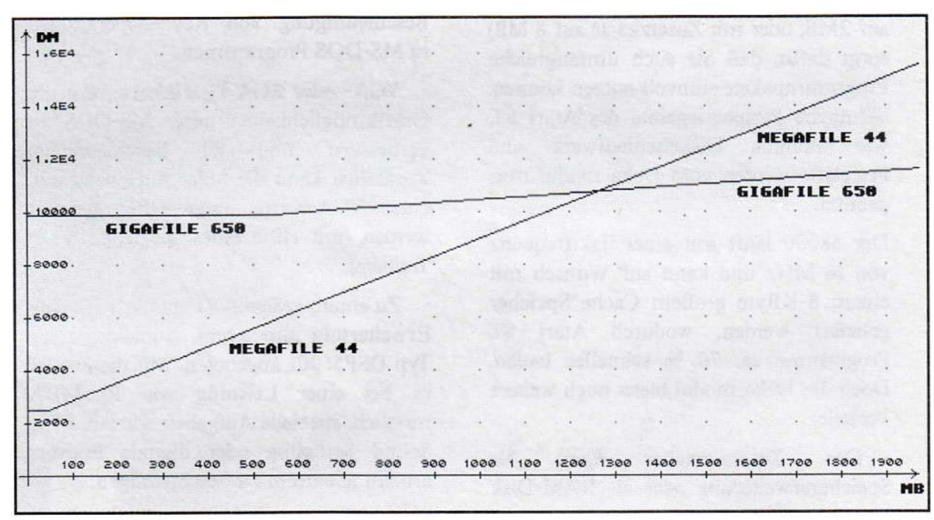



*Bild 5: Vergleich der Kosten für Speicherplatz bei MEGAFILE 44 und GIGAFILE 650*

Die Tabelle 2 zeigt, wie die GIGAFILE 650 in den beiden anderen Benchmark-Disziplinen (BIOS- und GEMDOS-Test) abschnitt. Dabei fallen wiederum zwei Besonderheiten auf: Der BIOS-Transfertest liefert für die CSS GIGAFILE leicht höhere Werte als das hardwarenahe CHECKHD (Tabelle 1) - hier lief das Benchmark-Programm offensichtlich dem Hardware-Cache des SCSI-Controllers in die Falle. Das zweite wichtige Ergebnis: Das aufwendige Schreibverfahren schlägt beim Anlegen und Löschen von Dateien heftig durch - beim GEMDOS-Lesetest schlägt sich die GIGAFILE dagegen wacker. Für den Chronisten: Alle Tests liefen auf einem MegaST 2 unter TOS 1.4.

## Vielzüngiges Schriftentum

Zur Zeit des Tests war das Handbuch zur Platte nicht fertig; ich konnte nur ein paar Calamus-Seiten begutachten. Jetzt, da Sie diesen Test lesen, sollte ein Handbuch fertig sein, das in deutsch, englisch und französisch auf jeweils etwa zwanzig Seiten den Umgang mit der GIGAFILE erläutert - mit Schwerpunkt auf Anwenderseelsorge, so CSS. Ob das gelingt, kann ich bis dato nicht beurteilen.

Ein weiteres Imponderabilium war zur Testzeit noch die Software: Zum GE-SOFT-Host-Adapter liefert die Firma GESOFT zwar auch Software mit (Treiber, Installationsprogramm, Cache-Programm), doch war diese noch nicht auf die GIGAFILE abgestimmt; beim Test behalfen wir uns mit einer gepatchten HDX-Version, dem ATARI-Treiber und einer Betaversion des CBHD, was auch wunderbar klappte. Beim Herumspielen mit dem Laufwerk fielen aber Merkwürdigkeiten auf, die es Plattenprogrammen etwas schwierig machen: Die Rückgabeparameter des MODE-SENSE-Kommandos schienen mir etwas rätselhaft, was aber wohl hauptsächlich an der fehlenden Dokumentation zum Sony-Controller liegt. SED und GIGAFILE verstanden sich jedenfalls nicht immer, ähnlich ging es anderen sehr plattennahen Programmen (das sind wenige).

Das braucht Sie als potentiellen Anwender nicht zu schrecken: CSS kündigte Besserung bei der Software an; man will ein einfach zu bedienenden Installationsprogramm sowie einen Treiber zur Verfügung stellen. Genaueres steht dazu im Moment (Januar) noch nicht fest. Man frage also vorher nach.

## Preisfragen

CSS bietet die GIGAFILE 650 in einem bis 31. März gültigen Einführungsangebot für 9980 DM an. Für seine zehn Riesen bekommt man das Laufwerk, das Handbuch, ein Medium, den Host-Adapter für den ST samt Software sowie ein SCSI-Kabel zum Anschluß an den TT - optimistisch gedacht, CSS! (Ob sich ATARI dadurch angespornt fühlt? Schaden kann's jedenfalls nicht...) Die GIGAFILE ist auch in einer Version für den PC erhältlich.

Der Preis ist für optomagnetische Laufwerke relativ günstig - aber eine Allerweltsanschaffung ist die GIGAFILE 650 eben doch noch nicht. Trotzdem kann sich ein optomagnetisches Laufwerk für Anwendungen lohnen, bei denen man immense Datenmengen umherschieben muß, die dazu noch sicher untergebracht werden sollen. Man denke an DTP-Programme, deren Font- und Bildersammlungen sehr schnell eine Platte füllen, an CAD-Anwendungen, die ebenso verschwenderisch Platz zu verschlingen pflegen, oder gar an speichergefräßige Sampler-Anwendungen. Den hohen Anschaffungskosten stehen relativ geringe Folgekosten pro Megabyte entgegen: Ein Medium kostet 698 DM, man kommt also auf 1.17 DM pro Megabyte. Zum Vergleich: Das Medium der MEGAFILE 44 kostet 298 DM und bietet knapp 44 MB, womit wir auf etwa 6.77 DM pro Megabyte kommen. Daraus folgt, daß sich eine GIGAFILE 650 schon ab einem Speicherbedarf von knapp 1300 MB lohnen kann - siehe Bild 5 (dort wurde ein Preis von 2498 DM für eine MEGAFILE 44 samt Medium veranschlagt, was in der Praxis bereits etwas unterschritten wird). Bei CSS plant man darüber hinaus, nicht nur die Medien mit 512-Byte-Sektoren einzusetzen, sondern auch die 1024-Byte-Sektoren für den ST nutzbar zu machen. Datentransfer und Kapazität erhöhen sich damit um jeweils 10%, die GIGAFILE 650 lohnt sich dann schon unterhalb der 1300-MB-Schwelle.

Ab dem dritten optischen Medium liegen die Kosten pro Megabyte beim optischen Laufwerk unter denen einer Wechselplatte. Wer voraussichtlich mit derart großen Datenmengen umgehen muß, wird sich also über diese Erweiterung des Massenspeicherangebots freuen. Allerdings muß man immer bedenken, daß ein optisches Laufwerk zur Zeit noch nicht die schnelle Platte für die tägliche Arbeit ersetzen kann - dazu ist die neue Technik doch noch etwas zu träge.

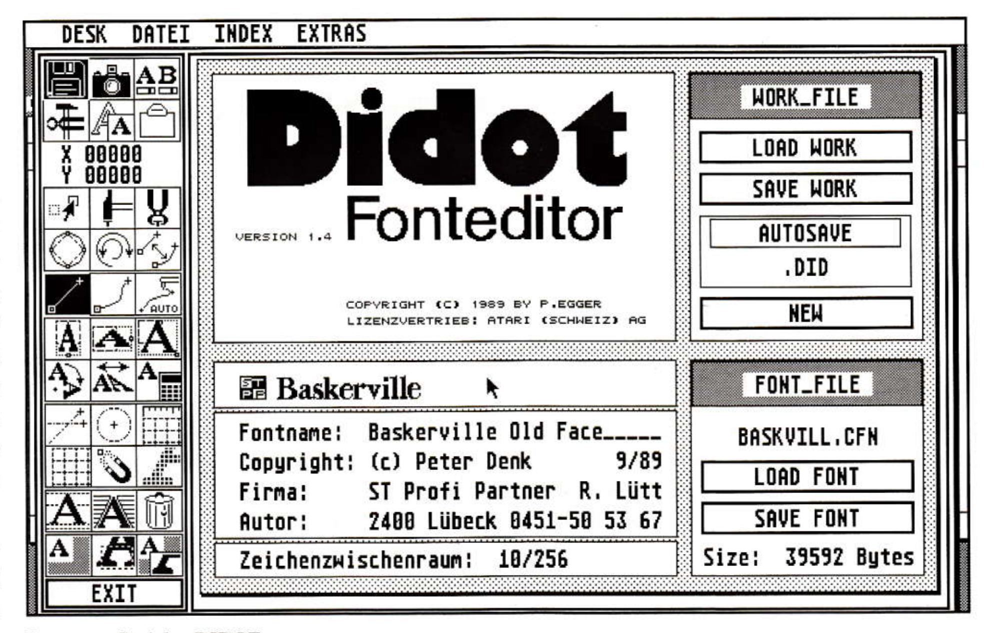
*CB*

***Bezugsadresse:***

*CSS Computersysteme Suplie*
*Landwehr 53*
*Postfach 1970*
*D-4670 Lünen*
*Tel. 02306/52489*

# fontastico DIDOT

**Ein neuer Font-Editor für Calamus-Fonts aus dem Lande der Eidgenossen schickt sich an, dem DMC-Font-Editor Konkurrenz zu machen, und nicht nur das - er erhebt den Anspruch, einen Standard zu setzen. So wie einst der Pariser Schriftgießer Francois Ambroise Didot und dessen Sohn Firmin Didot, die 1785 das typographische Maßsystem neu ordneten und den heute noch gebräuchlichen typographischen Punkt (Cicero) schufen ( 1 Pt. Didot = 0,376 mm).**




*Das neue Gesicht: DIDOT*

## Der Anspruch und die Tatsachen

Also nicht von ungefähr wurde gerade der Name Didot für den neuen Font-Editor gewählt, der jetzt auch von ATARI Deutschland vertrieben wird. Aber kann das Produkt dem Anspruch auch gerecht werden? Wird tatsächlich ein neuer Standard gesetzt in der Arbeitsweise? Ich meine ja. Wer mit dem DMC-Font-Editor gearbeitet hat, wird an vielen Stellen Möglichkeiten der GEM-Oberfläche vermißt haben, die die Arbeit erleichtern.

Wer zudem noch die 19"-Version des DMC-Editors unter Protos benutzt hat, um einen 19"-Monitor auf dem SM 124 zu emulieren, wird über der Arbeit manchmal verzweifelt sein. Protos bremst bei einem Bildschirm von ca 800 x 800 Punkten die Geschwindigkeit so stark ab, daß das ständige Neuzeichnen der ganzen Kontur bei Setzen oder Verschieben eines Punktes die Arbeit quälend langsam macht. Mit Didot braucht man weder einen echten noch einen emulierten 19"-Monitor, weil die eingebaute Lupe so flexibel ist, daß man völlig flüssig arbeiten kann.

Bei Didot kann einfach mit der gedrückten rechten Maustaste der Bild-ausschnitt verschoben werden - auch in der Lupe. Ein gescanntes Bild im Hintergrund läßt sich davon unabhängig verschieben! Man ist also nicht wie im DMC-Editor darauf angewiesen, beim Ausschneiden einer Vorlage sofort 100prozentig den richtigen Ausschnitt zu finden (Ich hatte in meinem letzten Artikel beschrieben, wie man mit einem Hilfsrahmen aus Linien im Calamus den richtigen Ausschnitt treffen kann beim Snapshot im DMC-Editor).

Didot kann auch einen Ausschnitt aus einem großen IMG-Bild frei verschieben. Ein weiterer Vorteil von Didot ist, daß gescannte Hintergrund-Bilder mit zunehmender Vergrößerung nicht zunehmend "dünner" werden, weil die Rasterpunkte auseinanderrücken - auch in der größten Vergrößerung ist die Vorlage immer klar erkennbar. Aber auch viele kleine Detailverbesserungen gegenüber dem alten Standard bei gleichartigen Funktionen machen die Arbeit angenehm.

Beim Kerning beispielsweise kann man in Didot einen ganzen Zeichensatz "durchrauschen" lassen mit einem Mausklick, während bei DMC die Bearbeitung eines Zeichensatzes Buchstabe für Buchstabe zur "Klickeritis" oder zur Verwendung des GEM Retrace-Rekorders führte (ein Hilfsprogramm, das Mausbewegungen aufzeichnet und abspielt).

Ein weiterer dicker Pluspunkt ist die wirklich GEM-mäßige Arbeitsweise mit Didot: Verzerrungen und Drehungen von Buchstaben sind über eine Gummiband-Box mit Handle äußerst einfach möglich mit optischer Kontrolle in Echtzeit (im Gegensatz zum DMC-Taschenrechner, der nach Tastatureingaben vor sich hin rechnet ...).

## Das i-Tüpfelchen ...

Allerdings wäre es das i-Tüpfelchen für Didot, wenn man eine gefundene Einstellung genauso auf einen ganzen Zeichen-

satz anwenden könnte wie beim Kerning. Was die Wiederholbarkeit von Verzerrungen betrifft, hat DMC mit den Zahlenwerten Pluspunkte, sei es über den Taschenrechner oder über das M-Square, mit dem man Schriften in Breite und Höhe verändern kann.

In Didot kann man sich jedoch mit Winkelangaben, Koordinaten und Hilfslinien helfen und die Vorteile der optischen Kontrolle genießen. Auch die Funktion "Zeichensatz zuladen" von DMC hat Didot nicht vorzuweisen, was besonders bei Sonderzeichen schade ist, die man bei DMC einfach auf freie Positionen zuladen kann. Dafür ist bei Didot die Clipboard-Funktion wesentlich besser und leistungsfähiger gelöst durch fünf (!) freie Clipboards und die jederzeit gegebene optische Kontrolle über die Verschiebung von Linienzügen oder ganzen Buchstaben. Auch das Springen zum nächsten Buchstaben geht bei Didot schneller und einfacher.

Außerdem ist es nach Studium des Handbuches bei Didot ganz einfach, Linienzüge an bestimmten Stellen mit einer Schere aufzutrennen oder mit einer Schusterahle zusammenzufügen (Alternate und linke Maustaste bei Hammer und Zange).

Beide Editoren haben ähnliche Funktionen mit ähnlichen Icons, aber die Umschaltung der Lupen z.B. geht bei Didot blitzschnell; man sieht auch gleich optisch, welche Linien man in den Mülleimer zerrt usw., die Verschiebung der Hilfslinien geht einfacher, und man kann auch Kreise als Hilfslinie generieren.

## Zukunfts-Bonbons

Einen Taschenrechner wie bei DMC hat Didot zur Zeit nicht, das vorhandene Icon ist vorgesehen für spätere Erweiterungen, genauso wie ein Icon, das automatische Vektorisierung von Hintergrundbildern verspricht.

Ein weiteres Zukunfts-Bonbon von Didot sind Zusatzmodule, die die Umwandlung von Calamus-Fonts in Postscript-(!), Signum-, GEM-DOS- und Ultrascript-Fonts ermöglichen. Die damit gegebenen Möglichkeiten sind gar nicht absehbar ...

Allerdings wird es noch eine Weile dauern. Entweder der Treiber GDOS.CNV wird vom Programm nicht gefunden (Vers. 1.4), oder er produziert nur Müll, der nicht als GEM-DOS-Font zu laden ist (Vers.1.3).

Aber es ist ja noch nicht aller Tage Abend, und der Gedanke, nebenbei auch PostScript- und andere Fonts zu erhalten, beflügelt die Ausdauer des einsamen Schriftenschnitzers ...

## Der erste Versuch - Verwirrung

Generell unterscheidet sich die Arbeitsweise bei beiden Editoren beim Zeichnen der Kurven: Bei DMC werden Punkte gesetzt, und die Tangenten müssen danach entfaltet werden (wir erinnern uns, Tangenten sind die Linien, die die Kurve tangieren, d.h. berühren). Bei Didot sind die Tangenten gleich "ausgefahren",aber nur ihre Endpunkte als Kreuze sichtbar, und man kann sie auch nicht als Linie sichtbar machen wie bei DMC.

Bei meinem ersten Versuch mit Didot herrschte also spontan Verwirrung, und zwar doppelte: Die Kurven schlängelten sich ungeachtet meiner gescannten Hintergrundbuchstaben, und ich wußte erst einmal gar nicht, welches Kreuzchen zu welcher Tangente gehörte ...

Nach einigen Versuchen wurde aber schnell das System klar: Didot ordnet die Tangenten automatisch so an, daß Ein- und Ausfallwinkel bei einem Punkt gleich sind. Bei Kreisen ist dann der Kreis auch gleich perfekt, wenn die Punkte richtig gesetzt sind. Bei unregelmäßigen Kurven müssen die (richtigen) Kreuzchen an die richtige Stelle geschoben werden.

Sieht ein grob gezeichneter Buchstabe bei DMC (ohne "Entfaltung" der Tangenten) erst einmal hölzern und eckig aus, so wirkt er bei Didot meist wie eine verrückte, unregelmäßige Pinselschrift. In beiden Fällen müssen die Endpunkte der Tangenten richtig verschoben werden, und dann stimmt's - nur: bei Didot kann man sehen, wie die Kurve sich verbiegt, während man bei DMC den Punkt neu setzt und wartet ...

Nach diesem Blick in zukünftige Weiten zum Naheliegenden: Meist wird nicht gleich ein ganzer Zeichensatz angefertigt, sondern ein dringend benötigter Schriftzug oder ein Logo soll "mal eben schnell" erstellt werden.

## Hier einige Tips zum Vorgehen:

*1) Auswahl der Vorlage:* Nehmen Sie die besten Vorlagen, die Sie bekommen können, auch wenn es etwas umständlicher ist, weil evtl. mal etwas herausgesucht werden muß aus alten Schriftmuster-Katalogen (wo die Alphabete noch größer abgedruckt waren als heute) oder mal kurz abgesetzt werden muß auf einer anderen Anlage. Ich habe schon Calamus-Fonts gesehen, bei denen offensichtlich ein Signum-Font als Vorlage gedient hat und nicht das ursprüngliche Original der Schrift. Daß dabei keine Ähnlichkeit mit dem Original herauskommt, wenn ich es auf wenige Pixel in Höhe und Breite zerhacke, sollte klar sein. Also: saubere Vorlagen, Schriftgröße um 36 Punkt oder größer (Didot kann noch zoomen in der Kamera-Funktion).

*2) Art der Wiedergabe:* Didot kann fertige Degas- und IMG-Bilder laden (von IMGs auch Ausschnitte verschieben). Hier sollte man die Stellung der gewünschten Buchstaben auf dem Bild berücksichtigen, damit genug Platz zum Verschieben ist. Die Bilder können mit Scannern oder Snapshots erzeugt werden.

Didot kann aber auch selbst als Accessory aus jedem Mal- oder Zeichenprogramm (auch aus Calamus) Snapshots machen. Auch hier kann man Hilfslinien für Schriftlinie, Oberkante von Groß- und Kleinbuchstaben gleich ins Bild zeichnen. Die Verwendung von Calamus als Vorlage für den Snapshot hat den Vorteil, daß über "Einstellbare Größe" und "Optimieren für Bildschirm" die Schrift in der Größe gut angepaßt werden kann (Einstellung für später aufschreiben).

*3) Positionierung:* Da bei Didot Hinter- und Vordergrund (auch in der Lupe) gegeneinander verschoben werden können, und zwar schnell und unter voller optischer Kontrolle (!), ist die Positionierung kein Problem: Anhand der vorher eingezeichneten Hilfslinien sieht man dann auch, um wieviel z. B. ein O oder V über die Schriftlinie hinausgeht, und berücksichtigt das. Beim Kerning hat Didot übrigens noch ein Schmankerl in puncto Positionierung parat: Wenn man Block-Kerning oder automatische Optimierung auslöst, wird automatisch der ganze Buchstabe auf Mitte der Zeichenfläche gerückt!

*4) Das Zeichnen der Kontur:* Auch bei Didot ist es am einfachsten, wenn man die Scheitelpunkte der Kurven sucht und dort die Punkte setzt. Dann kann man die Tangenten genau senkrecht oder waagerecht ziehen und gut kontrollieren, ob bestimmte Linien wirklich parallel laufen (die Tangenten müssen dann auch parallel

sein). Das klappt bei fast allen Kurven, nur bei längeren komplizierten schrägen Kurven (Schreibschriften) reichen die Scheitelpunkte nicht aus, man muß sie durch Verbindungspunkte ergänzen, deren Tangenten dann auch schräg laufen müssen. Das ist aber die Ausnahme. Die regelgerechten Scheitelpunkte findet man ganz einfach an den zeilenweise gerasterten Hintergrundbildern, z. B. der obere Scheitelpunkt eines "O" liegt genau in der Mitte der obersten Pixelzeile ... Außerdem hat Didot noch Funktionen parat, um Punkte nur senkrecht oder waagrecht oder in einem bestimmnten Winkel zueinander zu setzen, mal abgesehen vom ebenfalls vorhandenen magnetischen Hilfsraster.

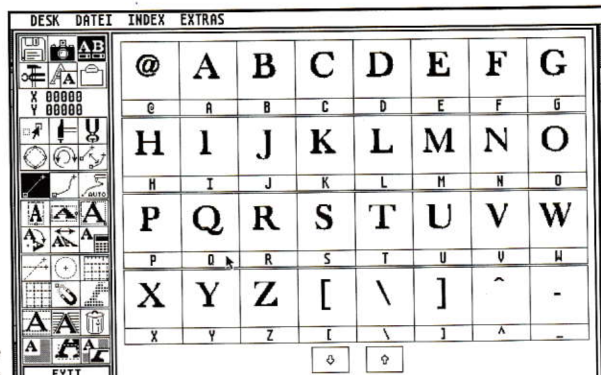
*5) Kombination von Konturen*: Genau wie beim DMC-Editor muß man ab und zu mal die Kontur einer Innenfläche in der Drehrichtung ändern, damit die Innenfläche auch weiß wird. Das geht über das entsprechende Icon und kann mit Hilfe der Kerning-Funktion kontrolliert werden. Wenn man Linienzüge "aneinanderflickt", dürfen sie nur einen Anfangs- und einen Endpunkt haben. Sonst erhält man nie die gewünschten Buchstaben. Aber dafür gibt es ja die "Ahle". Auch wirkt es bei der Verwendung der erstellten Buchstaben in OUTLINE von DMC störend, wenn Buchstaben aus mehreren Stücken zusammengesetzt sind. Man sieht dann nämlich die Nahtstellen, wenn man die Schrift als Outline darstellen läßt. Das ist der Fall, weil Outline die Teile als eigenständige Vektorgrafiken ansieht. Darum ist es besser, die Buchstaben "aus einem Stück" zu schnitzen, zur Not kann ja wieder die Ahle benutzt werden.

## Wer mehr vorhat ...

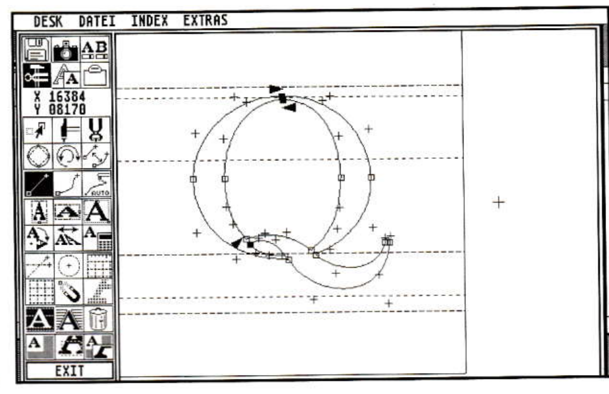
Die nun folgenden Hinweise beziehen sich auf das Erstellen ganzer Schriften.

*6) Kerning:* Was Kerning ist, wird ja im Calamus-Handbuch erklärt. Hauptziel des Kernings sind: a) gute Lesbarkeit und b) gleichmäßige Grauwirkung einer Zeile oder eines Textblocks. Man kann gut mit der automatischen Kerning-Optimierung in Didot arbeiten, wenn man sie als Grundlage sieht für die unbedingt notwendige manuelle Nachbearbeitung.

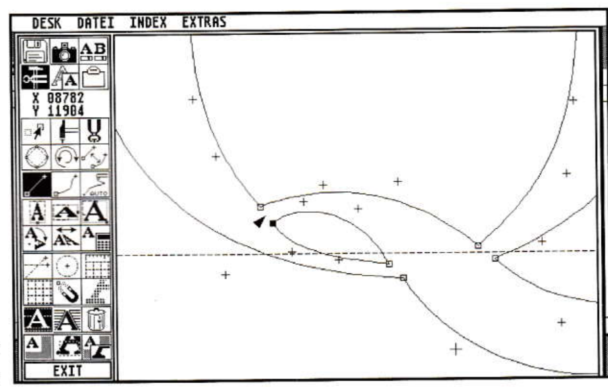
*Vorgehen:* Man erstelle sich ein Calamus-Dokument, in dem man in 10 oder 12 Punkt Größe einen längeren Text mit dem neuen, automatisch im Kerning optimierten Zeichensatz ausdruckt. Am besten trennt man den Text in Calamus mit der Funktion "Einfügen von Trennfugen" im Block oder beim Importieren, damit man ausreichend kleine Wortzwischenräume erhält. Dann sieht man auch genau, wo in den Wortbildern Löcher und Klumpen entstanden sind. Nun muß man anhand verschiedener Buchstabenkombinationen herausfinden, wer der Hauptschuldige ist, der linke oder der rechte Buchstabe an einem Loch/Klumpen. Das Abtippen aller möglichen Buchstabenkombinationen ist nicht der Weisheit letzter Schluß - entscheidend ist die gleichmäßige Grauwirkung, wenn man mit leicht zugekniffenen Augen einen Textblock betrachtet.



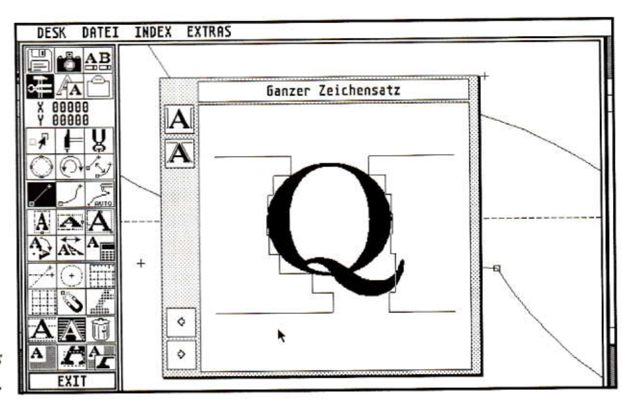

*Die schnelle Auswahlbox*



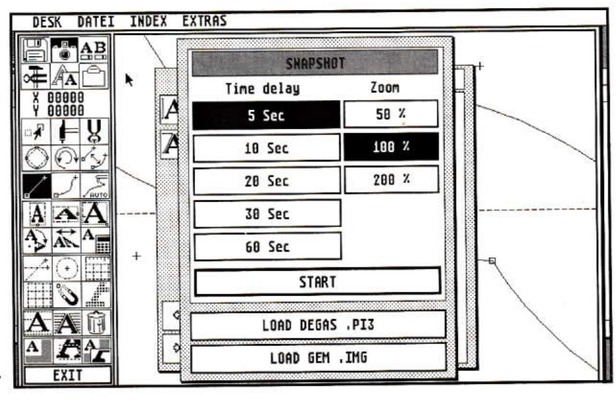

*Die Arbeitsfläche...*



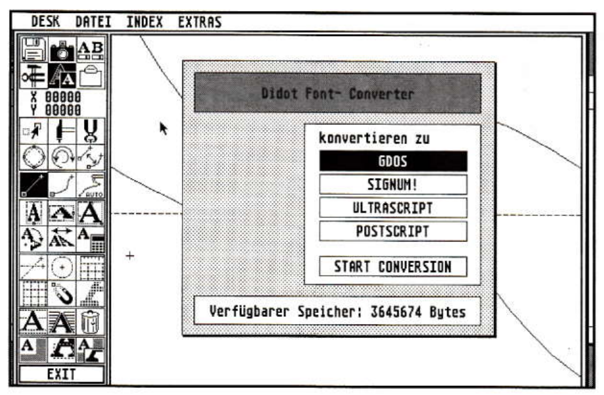

*... und die rasante Lupe*



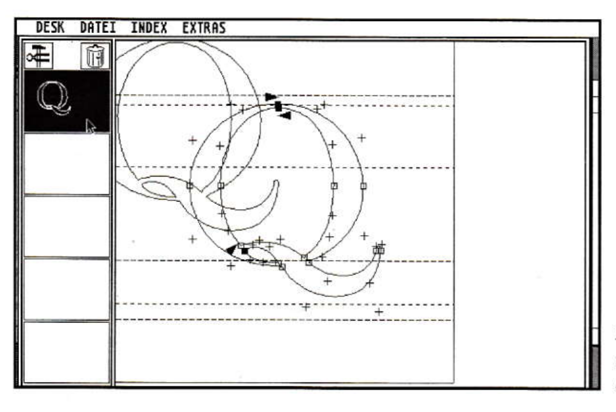

*Hauptsache, das Kerning stimmt...*



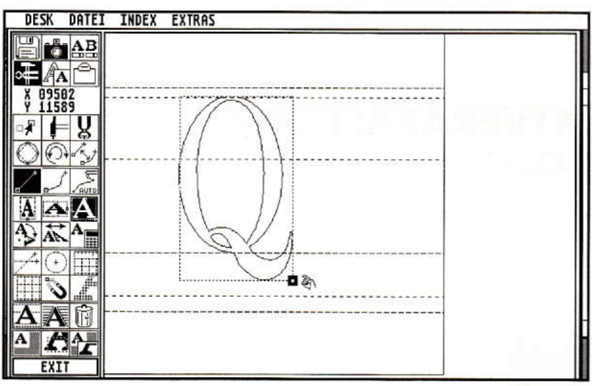

*Snapshot mit Pfiff*



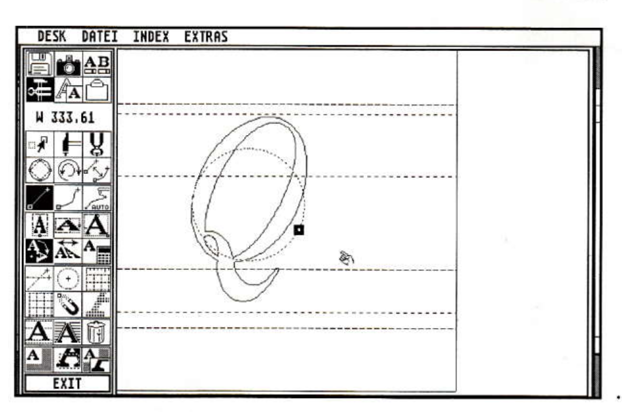

*Bonbon für die Zukunft - Konvertierung zu PostScript*



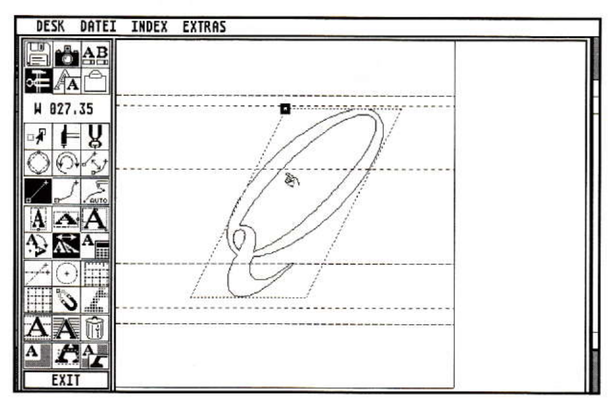

*Fünf Clipboards und volle optische Kontrolle*




*Verzerrung...*




*...deluxe...*




*... und realtime*

Bei Calamus-Fonts ist, anders als bei GEM- oder Signum!-Fonts, die Phantasie des Schriftschnitzers achtfach gefordert, weil der ja nicht nur eine Buchstabenbreite festlegen muß, sondern acht Zonen übereinander (ja, es sind acht, nicht sieben, wie in meinem letzten Artikel steht).

Man muß einfach mal die DMC-Schriften Times und Swiss im Kerning betrachten, dann bekommt man schon Anregungen positiver und negativer Art. Man sollte sich Zeit nehmen für das Kerning, mindestens so viel wie für das Zeichnen der Konturen(!). Auch muß die Eigenart einer Schrift bedacht werden, ob es eine Schrift mit oder ohne Füßchen ist, eine Schreibschrift oder eine Fraktur. Dieses Thema könnte man in einer weiteren Folge zum Font-Editor aufgreifen, da es sehr umfangreich ist.

Hilfreich ist Didot auch beim Kerning von Hand, weil die Verschiebung der Kerning-Zonen glatt, flüssig und exakt geht im Vergleich zu dem hakeligen Geklicke bei DMC. Auch braucht man nicht zwischen linkem und rechtem Rand umzuschalten, man packt einfach mit der linken Maustaste zu und zieht.

*7) Font-Name/Pixel-Bild:* Den Font-Namen trägt man am besten erst zum Schluß ein, wenn alle Buchstaben vorhanden sind, die die Pixel-Routinen brauchen, um ein grobes Abbild in der neuen Schrift zu erstellen (Sie wissen schon, der Schriftzug, der nachher in Calamus im Textmenü zeigt, welche Schrift man geladen hat). Hier sind die Ergebnisse etwas schlechter als bei DMC, wahrscheinlich aufgrund anderer Routinen zur Umformung. Das trifft auch auf die Darstellung im Buchstaben-Auswahl-Menü (Setzkasten) zu, die bei DMC schöner ist. Dafür geht das Scrollen und Auswählen aber bei Didot schneller, und entscheidend ist das Druckergebnis.

*8) Export:* Seit der neuesten Version Didot 1.4, die mir von ATARI zur Verfügung gestellt wurde, besteht sogar die Möglichkeit, den Inhalt der Zeichenfläche als Calamus-Vektorgrafik (CVG) zu exportieren. Das öffnet natürlich die Tür zu Zeichenprogrammen, die CVG importieren (und konvertieren) können und damit zu anderen Layout-Programmen oder z.B. eine Seitentür zu Outline. Leider kann Vektorgrafik noch nicht in Didot importiert werden, denn dann könnte man ...

## Fazit

Unter dem Strich sprechen eine Vielzahl von Punkten für Didot. Der Anspruch, einen neuen Standard zu setzen, ist erfüllt worden, und es liegt nur an den fehlenden i-Tüpfelchen, daß man in naher Zukunft gut damit beraten ist, beide Font-Editoren zu haben (automatische Verzerrung über M-Square, Zuladen und andere Funktionen bei DMC). Für den Anwender ist es gut zu wissen, daß Konkurrenz herrscht und der Standard für Calamus-Font-Editoren wohl auch in Zukunft weiter verbessert wird. Fontastisch!

*Peter Denk*

*Bezugsadresse:*

*Alle ATARI-Fachhändler*

# NICBASE aus Holland

## Wie eine Schachdatenbank ein kariertes Schachspielerleben bunter gestalten kann



M.V. ZMMERMANN

**Die Geschichte des modernen Schachspiels ist kurz - knapp 200 Jahre sind vergangenen, seit der Franzose Philidor mit der Feldherrenmentalität vieler Vorgängergenerationen von Spielern brach und lehrte, den eigenen Verstand zu benutzen. Es war das Zeitalter der Aufklärung, und Philidor reformierte das Schachspiel von Grund auf.**

Die Entwicklung eines wissenschaftlichen Schachstils ging seitdem ebenso rapide voran wie der technologische Fortschritt. Heutzutage geschieht die Spielanlage unter logischen Gesichtspunkten, das Schach wird nicht mehr als Einheit, sondern als integrativer Aufbau mit den drei Spielabschnitten Eröffnung, Mittelspiel und Endspiel begriffen. Da sich in der Vergangenheit immer mehr aktive Schachspieler in Vereinen und Verbänden zusammenschlossen, wurde Schach schließlich zu einem Weltspiel, das in der Entspannungsphase nach dem Kalten Krieg sogar durch den Wettkampf UdSSR gegen eine Weltauswahl und den Weltmeisterschaftskampf Spassky gegen Fischer politische Geltung erhielt.

### Schach dem Computer

Seit Beginn des achten Jahrzehnts dieses Jahrhunderts steht Schach aber nicht nur als Chiffre für den intellektuellen Kampf zweier Menschen. Der Computer, seit jeher ein begabtes Wesen, bei der Verwaltung von Zahlen und Buchstaben Erstaunliches zu leisten, hat eines seiner ersten Anwendungsgebiete in der Schachprogrammierung gehabt. Innerhalb weniger Jahre wurden Schachcomputer programmiert, die den meisten Vereinsspielern spieltechnisch überlegen waren und so manchen Internationalen Meister das Fürchten lehren konnten und können.

Für alle, die ernsthaft um den Tribut aus Caissas Waagschale ringen, dürfte das Schachspiel schon längst zur leistungsorientierten Pflichtübung gehören. Die Phase der Vorbereitung auf einen bevorstehenden Turnierkampf nimmt in den meisten Fällen mehr Zeit in Anspruch als die Partie selbst. Eröffnungswissen muß beständig aktualisiert, die neusten Großmeisterpartien nachgespielt und neue Systeme für eine effizientere Spielanlage durchdacht und einstudiert werden. Das Training des erfolgreichen Schachspielers grenzt an die Methodik wissenschaftlichen Arbeitens. Nicht selten sah man früher gelegentlich Kämpen mit Karteikarten in der Hand, die den notierten Lernstoff wie Vokabeln wiederholten. Der wissenschaftliche Charakter des modernen Turnierschachs eröffnet dem Fleißigen den Erfolg auf den vierundsechzig Feldern. Mit Intuition ist da nur wenig auszurichten, wenn das Gedächtnis nicht geschult und die Routine nicht professionell durchorganisiert ist. Für die zahllosen Schachspieler, die im Wettkampf an der Spitze mithalten wollen, ergibt sich allzu leicht bei den tausenden von in einem Jahr gespielten Partien eine nicht zu verkraftende Überlastung an Informationen respektive des aktuellen Theoriewissens.

Seit den siebziger Jahren gibt es Halbjahresbände von Partiesammlungen, die die wichtigsten Meisterkämpfe verzeichnen. Ihre ausgeklügelten Eröffnungsschlüssel und Bewertungssysteme stellten das Hilfsmittel der häuslichen Analysepraktik dar, bis jemand auf den fundamentalen Gedanken kam, daß der Computer, in diesem Falle richtungsweisend der ATARI ST, die verwaltungstechnischen Aufgaben einer Schachdatenbank übernehmen könne. Ein Hauptanwendungsgebiet von Schachdatenbanken ist die Verwaltung von Eröffnungsbibliotheken nach einem klassifikatorischen Schlüsselsystem und insbesondere die präzise Zuordnung von Partie und Eröffnungsschlüssel.

### Zwei Systeme

Vorab sollten Sie wissen, daß es in der Schachwelt zwei (beinahe) inkompatible Dynastien von Eröffnungsschlüsselsystemen gibt: einmal das ältere System des jugoslawischen INFORMATOR-Verlags und dann das niederländische NEW-IN-CHESS-Eröffnungssystem. Die seit zwei Jahren erhältliche deutsche CHESS-BASE-Schachdatenbank verwendet das

Desk File Info Position Key Play Game Base Gamefile
Abou Open Datab Set Up Sele Set O Load Find List in Gamefile
Close Gamef List Save Find in Gamefile
Desk Find Posi Find Remov Save Save
Desk Open Activ Remove Po Excha Load Save to Gamefile
Desk Close Key S Edit Shuff New G Load Next Selection
Desk Score New Key P Exam Strip Port
Desk Save Add Posit Play Fragm Selec Port Games to Base
Desk Acces Move Add Quiz Print Selected Games Info
Game Print Dia Move Print Remov Print Selected Games
Quit Game Result Sort Game Remove Game Remove Selected Games
Prin Analy Classify Ga
Remove Selected Keys Classify Gamefile

*Bild 1: NICBASE stellt sich vor. Eine Menüleiste voller Funktionsangebote.*

INFORMATOR-Schlüsselsystem. Sie sollten also, bevor Sie sich für den Kauf einer Schachdatenbank entscheiden, in Ihren Bücherschrank schauen und sich informieren, nach welchem Schlüsselsystem Ihre Partiesammlungen aufgebaut sind. Es spricht aber einiges dafür, sich mit NICBASE und dem NEW-IN-CHESS-SYSTEM anzufreunden. Doch lesen Sie selbst!

Wer mit NICBASE zusammenarbeiten möchte, sollte sich auf einen internationalen Standard gefaßt machen, denn er kauft ein niederländisches Produkt ein. Das Liefervolumen von NICBASE ist eher schmächtig zu nennen: eine doppelseitige Diskette, die in eine heftgroße Mappe eingelegt ist, und ein einhundertundelf Seiten umfassendes englischsprachiges Handbuch im DINA4-Format. Das deutschsprachige Benutzermanual ist angekündigt, aber auch die englischsprachige Fassung ist für jedermann mit rudimentären Sprachkenntnissen verständlich geschrieben und kann als problemlose Einführung in NICBASE gelesen werden. Das Programm wird von Diskette gestartet und betrieben, jedoch ist dies nicht ratsam wegen diverser Zeitverluste beim Nachladen von Programmteilen und Partiedaten. Deshalb empfehle ich auf jeden Fall (auch wenn meine ATARI-Harddisk irreparabel defekt ist und ihren Geist ausgehaucht hat), eine Festplatte oder zumindest, um die Laufzeiten für Ladeoperationen extrem zu verkürzen, eine RAM-Disk zu benutzen.

Eine feine Art des Kopierschutzes stellt die Indizierung der Software mit dem vollen Namen der Käufers sowie der Seriennummer dar. So muß der Lizenznehmer selber dafür Sorge tragen, daß seine erworbene Software mit den beiden Identifizierungsmerkmalen Name plus Seriennummer nicht in aller Welt als Raubkopie kursiert. In NICBASE ist ein "intelligentes" Installationsprogramm implementiert. Es tritt nur dann in Erscheinung, wenn mindestens einer der fünf Zugriffspfade nicht eindeutig definiert ist. Wenn die Arbeits-Floppy abweichend von Laufwerk A gewählt wird (C,D,F,... bei Hard- oder RAM-Disks müssen zuerst die Pfade der Datenbank neudefiniert sein), werden quasi fünf Leerdateien mit den Extenderkennzeichnungen .POS, .PTR, .KEY, .IND und .GAM auf das optionale Laufwerk geschrieben und dann von Diskette fünf volle Dateien mit den entsprechenden Extendern über die zuvor angelegten Leerdateien kopiert. Laufwerks- und Pfadänderungen werden wie immer auch hier mit einer File-Option/Save Environment abgespeichert.

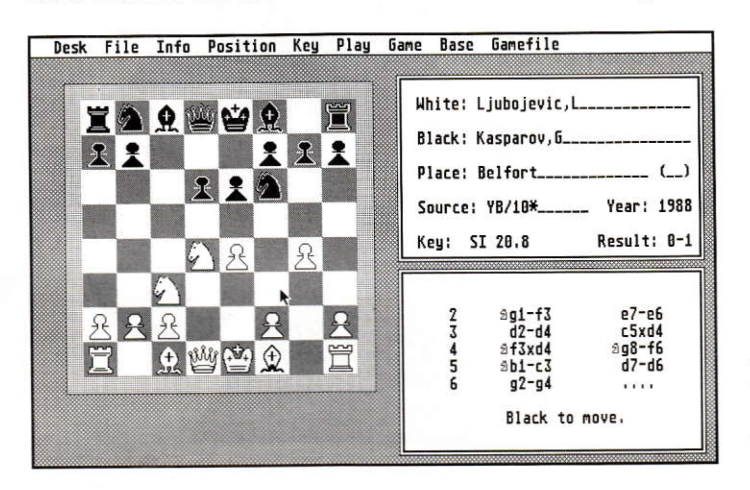



*Bild 2: Der Kommandobildschirm mit Partie-Info und Zugformular*

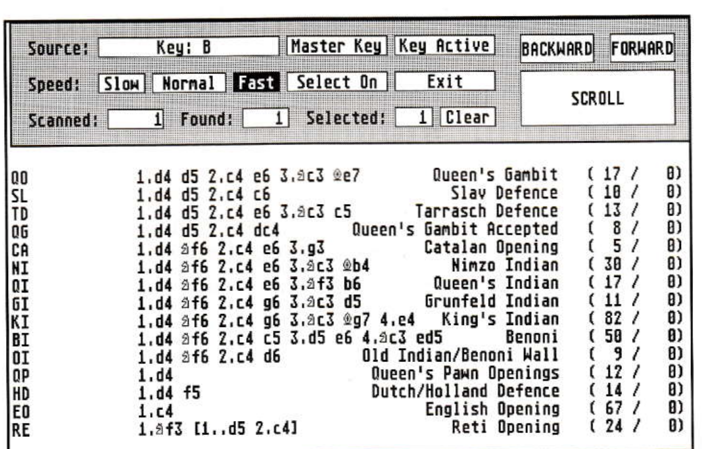



*Bild 3: Das Eröffnungsschlüsselsystem, das sämtliche Damenbauereröffnungen auflistet, inkl. Erwiderungen und analogen Zugfolgen (1.d4,c4,Sf3).*

## Die Datenbank

Nach Absolvierung der Installationbedingungen kann ich mich nun der Verwaltung von Schachdaten widmen. Zu jeder Arbeitssequenz gehört das sogenannte Game-File, in dem eine Auswahl von Schachpartien nach zeitlichen, räumlichen oder eröffnungsspezifischen Gesichtspunkten zusammengefaßt ist. Mir standen mit zwei Game-Files mehr als 1500 Partien zur Verfügung. So auch eine Kompilation sämtlicher von Robert Fischer bis 1972 gespielten Partien.

NICBASE gewährleistet, während einer Arbeitssitzung in mehreren Game-Files zu operieren und eine Partienauswahl nach individuellen Gesichtspunkten zusammenzustellen. Laut Handbuch können mit einer doppelseitigen Floppy, 1 MB RAM und einer 20MB-Harddisk bis zu hunderttausend Partien archiviert werden. In der aktuellen Version 1.1. werden fünfhundert Partien zur Selektion erlaubt. Die Beschränkung auf einhunderttausend Partien im GAMEFILE ist ein typisches Problem der ATARI-Hardware. Denn auf einer Partition ist die Höchstmarke an Speichervolumen auf 16 MB festgesetzt. Der Handbuchautor kommmentiert den technischen Sachverhalt kühl: "When this restriction disappears, this limit is also history."

Auch in den königlichen Niederlanden setzt man auf Statistik. So läßt sich mit 1040 Zügen (es sind wohl 720 Halbzüge gemeint?!) das längste Endspiel aller Zeiten aufnehmen. Mit möglichen zehntausend Stellungen wird eine Partieanalyse zur Präzisionsarbeit, die selbst höchsten intellektuellen Ansprüchen genügen dürfte. Des weiteren sind ca. fünfundsechzigtausend Positionen und Schlüssel innerhalb einer Datenbank möglich, wobei der Sammelleidenschaft und dem Informationsbedürfnis nach verzweigten Zugfolgen und verwickelten Eröffnungssytemen nur eine ungefähre Grenze im Rahmen der Speicherkapazität gesetzt worden ist. Ein besonderer Vorzug von NICBASE ist die listenweise Auswahl bestimmter Schachpartien per Mausklick.

Vor Aufnahme der Arbeit mit NICBASE sollte grundsätzlich überlegt werden, ob man mit einer einzigen großen Dokumentation oder mit mehreren kleineren Dokumentationen schlechter oder besser fährt. Es gibt jeweils gute Gründe, sich für eine der Alternativen zu entscheiden. Schachspieler A beispielsweise betreibt das Studium der vierundsechzig Felder semiprofessionell. Ihn interessieren nicht so sehr Mittel- und Endspielabwicklungen, sondern die Eröffnungsphase. Es ist ihm daran gelegen, möglichst komplett in wenigen Eröffnungsvarianten die gespielten Neuerungen kennenzulernen. Die Möglichkeit, ein großes Kontingent von Turnierpartien in einer Schachdatenbank archiviert zu halten, befreit diesen Spielertyp von dem mühlevollen Transkribieren relevanter Partien auf Zettel und Karteikarten. Sein Hauptinteresse liegt in der Partienanalyse, die meistenteils auf die ersten zwanzig Züge beschränkt ist. Soweit also, bis ein theoretisches Urteil über eine Partiestellung aus der Eröffnung gefällt werden kann. Das am Brett erzielte Resultat ist im Theoriefall allerdings nur sekundär interessant. Ein solcher - ich nenne ihn - “Eröffnungstheoretiker”, der sich vielleicht auf einen Wettkampf vorbereitet und dazu präzise Informationen über den, sagen wir, Keres-Angriff (6. ... g4) im Sizilianer benötigt und sich dann noch in seinen Partien mit Nimzo-Indisch und Französisch verteidigt, täte gut daran, möglichst kleine Dokumentationen anzulegen. So würden mit der Zeit sämtliche von einem Eröffnungstyp gespielten Partien modular in einer Dokumentation gesammelt werden. Mit der Zeit entstünde so eine kleine Geschichte einer Eröffnungsvariante. Die Option zum modularen Arbeiten mit NICBASE kommt der Spezialisierung auf Eröffnungssysteme sehr entgegen. Insbesondere Fernschachspieler werden die Option zur modularen Dokumentation zu schätzen wissen.

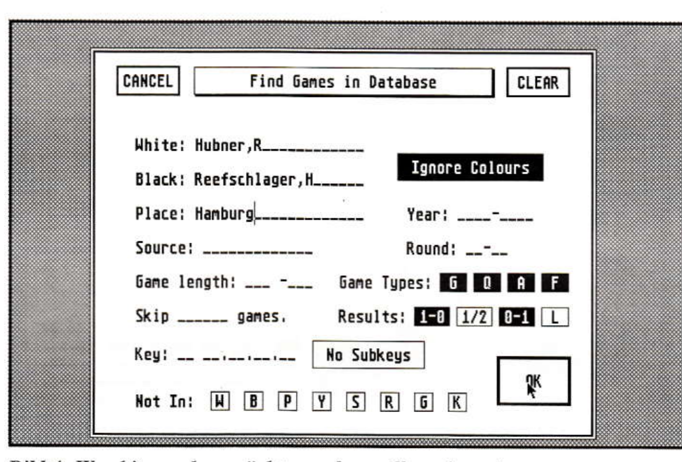



*Bild 4: Wer hier suchen möchte, muß vor allem ein gutes Namensgedächtnis haben.*

*Bild 5: Der Weg zum (fast) individuellen Datenbankdesign*

Doch es geht auch anders! Für die meisten NICBASE-Anwender wird es zweckmäßiger sein, mit einer einzigen Dokumentation zu arbeiten, in die - soweit die Speicherkapazität reicht - sämtliche Partien aufgenommen werden. Meistenteils sind die importierten Partien entsprechend dem NICBASE-Klassifizierungssystem unter einem Eröffnungsschlüssel archiviert. Anderenfalls erscheint in der Listenübersicht des Such- und Auswahlmenüs vor jeder aufgelisteten Partie der Hinweis “not classed”. Die Klassifizierung, d.h. die Beschreibung und Anpassung einer Partie an das NICBASE-Eröffnungsschlüsselsystem, erfolgt im Menü Gamefile/Classify Gamefile, und die codierte Partie wird unter dem identifizierenden Eröffnungsschlüssel archiviert. Von nun an finden sich ihre Kenndaten (beide Spielernamen, Jahr, Ort, Ergebnis) im Verzeichnis der betreffenden Eröffnung. Sie kann von dort aufgerufen werden.

## Wer die Wahl hat...

Bevor man eine Schachpartie überhaupt auf dem Bildschirm zu sehen bekommt, muß sie im Selektionsmenü ausgewählt werden. Nach dem Starten von NICBASE ist das Selektionsmenü quasi die erste Anlaufstelle, das Schaltzentrum, um die Schachdatenbank für die bevorstehende Arbeit zu präparieren. Wiederum gibt es zwei Wege: zum einen die Partie(n)-auswahl per Eröffnungsschlüssel, d.h. Partien einer Variante werden zum Nachspielen reserviert, zum anderen die Auswahl von Partien per Suchmanöver. Man durchforstet die Datenbank dabei nach bestimmten Spielernamen, Paarungen, Ergebnissen oder Turnierorten, wobei eine Kombination der Suchaufträge zur Selbstverständlichkeit gehört.

Für die Suche per Eröffnungsschlüssel sowie nach Positionen stehen eigene Suchmenüs zur Verfügung. In jedem Fall muß, bevor mit irgendeiner Schachpartie etwas begonnen werden kann, ein Suchauftrag erfolgen. Wenn der Suchauftrag nicht-leer ist, also mindestens eine Partie ausweist, wird dieser in einem Zwischenspeicher reserviert, der auch dann nicht gelöscht wird, wenn weitere Suchaufträge erfolgen. Für die Suche per integriertem NICBASE-Schlüsselsystem gibt es ein spezielles Kontrollfenster mit einigen interessanten Funktionen. Um mit Partien arbeiten zu können, wird eine Auswahl, entsprechend der Formulierung des Suchauftrages, in den Zwischenspeicher "gescanned". Die Auswahl von Partien über den integrierten Eröffnungsschlüssel erfolgt im Menü KEY, Option SELECT

KEY. Das hierarchisch aufgebaute System des Eröffnungsschlüssels wird aufgerufen. Er folgt einem Top-Bottom-Entwurf. Man muß sich von der Spitze zu der Eröffnungsvariante herabfragen, die schließlich analysiert werden soll. Die Abfragezeile enthält auf oberster Ebene die Kennzeichungen A und B als Alternativen, zwischen denen zu wählen ist. Nach dem Anklicken von A (für alle Varianten mit 1. e4 und Erwiderungen) oder B (für alle Varianten mit 1. d4, c4, Sf3 und alternative Eröffnungssysteme) erscheinen in Zeilen aufgelistet die Elemente des angewählten Eröffnungsschlüssels. Eine solche Zeile besteht an erster Stelle aus einem Kürzel der Eröffnung, dann folgen die identifizierenden Eröffnungszüge, der Name der Eröffnung und eine Doppelanzeige der Anzahl von archivierten und ggf. von ausgewählten Partien. Per Mausklick wählt man aus, mit welcher Eröffnung man es zu tun bekommen möchte. Es wird ein Schlüssel aktiviert. Alle in einem Schüssel enthaltenen Partien werden mit der entsprechenden Klassifikationsnummer aufgelistet. Jede Zeile enthält die folgenden Elemente: Klassifikationschiffre, Paarung, Spielort, Spieljahr, Ergebnis, Zügezahl. Per Mausklick sind einzelne Partien zu selektieren. Die angewählte Partie wird mit einem durchgehenden schwarzen Balken invertiert gekennzeichnet, und die Suche geht weiter.

Es gibt aber noch den Programmpunkt SELECT ON, mit dem Partien durch Bildschirm-Scrollen stapelweise in den Zwischenspeicher geladen werden können. Dann werden alle Partien ab der Stelle, ab welcher der Befehl gegeben wurde, invertiert dargestellt. Das Scrollen geschieht in drei Stufen. Der Mauszeiger muß ohne Tastendruck auf dem rechteckigen Scroll-Balken stehenbleiben. Die Geschwindigkeit wie die Laufrichtung müssen vom Menü aus gesteuert werden. Beim Umschalten von "Slow" auf "Fast" konnte ich keine erhebliche Geschwindigkeitssteigerung feststellen. Wenn größere Partieblöcke ohne Auswahloption überbrückt werden sollen, lohnt es sich, auf "Fast" umzuschalten.

Ansonsten bringt der Geschwindigkeitsvorteil keinen Nutzen. Denn es kommt ja auf die Lesbarkeit der vorbeihuschenden Partie-Infos an. So kann jeder sich fürs erste informieren, welcher Zündstoff in einer Datenbank enthalten ist und abschätzen, was für ihn interessant oder weniger interessant ist.

Wer mit NICBASE noch nicht allzu vertraut ist, wird zu Beginn seiner Schachstudien öfters diese Überblicksmöglichkeit nutzen, um an den heißbegehrten Stoff, aus dem vierundsechzig Felder sind, heranzukommen. Doch professionell ist das nicht. Eine Datenbank-Software effektiv

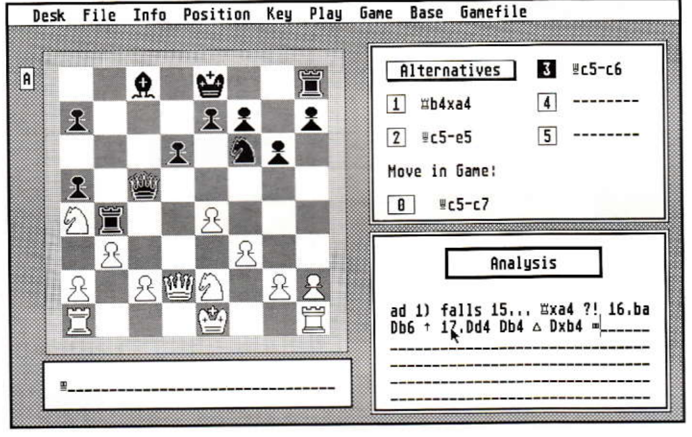



*Bild 6: Es schaut schon verführerisch aus! Und man kann wirklich jede Variante mit der verzweigtesten Abweichung rekonstruieren und dann auch noch fachmännisch kommentieren.*

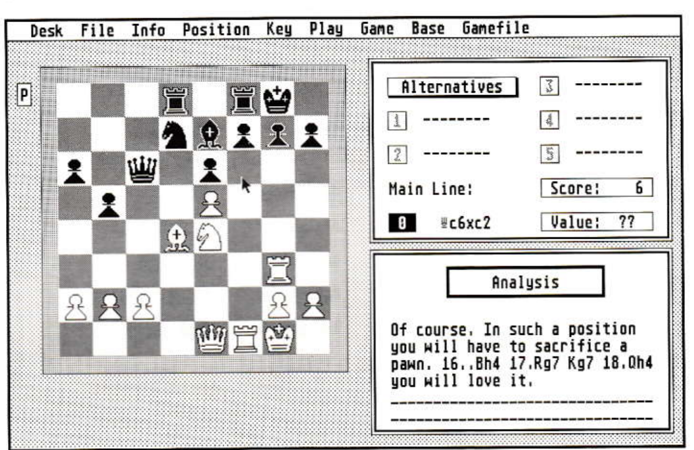



*Bild 7: Hätten Sie's gewußt? Wer ernsthaft punkten möchte, hat freie Hand zum eigenen Schachlehrbuch.*

zu nutzen, hängt insbesondere davon ab, wie das Suchmenü aufgebaut ist und wieviel Verknüpfungsebenen Suchaufträge zulassen. Bei einer Schachdatenbank, in der Stammdaten als Kombination von identifizierenden Personen- und Ortsangaben und schachtechnischem Aufschreibesystem abgespeichert werden, gehört eine sparsam haushaltende und trotzdem zuverlässig und schnell arbeitende Speicherverwaltung quasi zur Pflichtübung des Programmierens.

Wie Großstädte den Grad an Hektik der Menschen bestimmen, die in ihnen leben, so legt ein Computerprogramm seinen Benutzer auf ein bestimmtes Maß an Arbeitsgeschwindigkeit fest. Das ist mir besonders beim Suchmenü aufgefallen. Das integrierte Suchmenü kann von drei Programmpunkten her aufgerufen werden. In der Regel benutzt man aber im Menü BASE den Programmpunkt "Find Game in Base". Wer nicht gegen Windmühlen anrennen will (und welcher Schachspieler möchte das schon?), merkt hier recht schnell, daß NICBASE umfangreiche Kenntnisse in Caissas Paradies voraussetzt. Nicht nur die Bezeichnungen und die Stammzugfolge der gebräuchlichsten Eröffnungen müssen gekannt werden, sondern auch die Kenntnis der Namen von Meisterspielern und Turnierorten ist eine große Erleichterung. Beinahe ebenso wichtig wie eine Partiestellung zu analysieren, ist es zu wissen, welche Spieler für die geschehenen Züge einer Partie die Verantwortung tragen.

## Wann? Wer? Wo?

Am häufigsten wird man das Suchmenü für die Personen- oder Spielortsuche beanspruchen. Das aber wäre das Mindeste. Unbedingt sollte beachtet werden, daß ein Suchmanöver nach Spielernamen nur erfolgreich abgeschlossen werden kann, wenn Spielername und Vorname, durch Komma getrennt, angegeben werden. Da es umständlich ist, zu den meisten Namen der Koryphäen auch den Vornamen im Gedächtnis zu behalten, wirkt es sich negativ aus, daß keine Wildcards vergeben werden können.

Sie interessiert beispielsweise weder Eröffnungsstrategie noch Mittelspielgeplänkel vieler Schachpartien, sondern Sie möchten in die Theorie des Endspiels einsteigen. Da werden nur solche Partien für Sie interessant sein, bei denen sich die Gegner nicht nach spätestens vierzig Zügen die Hände geschüttelt haben. Im Suchmenü können Sie bequem das Partiestadium im Dialogfeld "Game length" auf mind. 40 - beliebig beziffern und erhalten so alle in der Datenbank auffindbaren

Endspielsituationen, die mit dem vierzigsten Zug beginnen. Sollten Sie den Suchauftrag noch weiter differenzieren wollen, so können Sie die Resultatsoption 1-0 hinzuschalten, und Sie erhalten sämtliche Endspielsituationen nach dem vierzigsten Zug, die von Weiß gewonnen wurden. Die ergänzende Kombination mit einem Spielernamen oder einer Paarung ist selbstverständlich. Nach jedem Suchauftrag lohnt es sich aber, mit CLEAR das Arbeitsmenü auf die Grundposition zurückzustellen, da sonst leicht Fehlmanöver auftreten. Mit OK bzw. dem RETURN/Enter-Befehl wird der Suchauftrag ausgeführt, d.h. die gesamte Datenbank wird nach entsprechenden Datensätzen durchsucht. Datensätze, die dem eingestellten Parameter bzw. einer Kombination genügen, werden im Zwischenspeicher für die Weiterverarbeitung reserviert.

Das federführende Menü für das Datensatz-Handling ist BASE. Mit dem Menüpunkt "Load Next Selection" werden die im Suchmenü ausgewählten Partien nacheinander aufgerufen. Aller Dienste des Suchmenüs darf sich auch mehrmals bedient werden, ohne daß eine Überspeicherung der älteren Fundsachen geschieht. Dem Anwender steht es damit offen, während einer Sitzung eine durch verschiedene Suchaufträge verzweigte, seinen Ansprüchen genügende individuelle Partiensammlung sich "zusammenzubasteln" und deren Inhalt auch separat abzuspeichern. Das Nachspielen einer aktuellen Partie erfolgt mit dem rechten Cursor-Pfeil oder, wenn das Notationsformular aufgeschlagen wurde, mit der linken Maustaste. Der Mauspfeil befindet sich dabei auf dem Formularblatt.

## Spielanalyse

Insbesondere verfügt NICBASE über ein umfangreiches Options-Menü, mit dem das Outfit des Desktops, falls Langeweile aufkommt, aufmöbliert werden kann. Mit oder ohne Notation, vier Hintergrunddarstellungen sowie die wahlweise Einstellung des GEM-Zeichensatzes machen für launenhafte Menschen die Arbeit mit der Datenbank abwechslungsreicher. Es läßt sich alles ein- und ausstellen, wobei zu berücksichtigen ist, daß die zweidimensionale Brettdarstellung jeweils die eine linke Hälfte des Desktops einnimmt, so daß maximal zwei Informationsanzeigen bzw. das Analyseformular genutzt werden können. Bei der Partie-Analyse muß man dann mitunter auf den Spielbericht verzichten, um dem Variantenreichtum einer Brettstellung auf die Spur kommen zu können. Falls im Option-Menü bei den Menüpunkten Alternatives, Analysis und Comment Line eine If-Auswahl getroffen ist, werden die Screens automatisch aktiviert bzw. deaktiviert, sobald eine Änderung der Analysesituation passiert. Den routierten Schachspielern darf gesagt werden, daß nach NICBASE Figuren und Bretter in den Nischen verstauben werden. Denn es wird eine vollkommen neue und äußerst komfortable Dimension der bildschirmgesteuerten Analysetechnik geboten. Das moderne Schach lebt, getraut man sich einen Blick in die einschlägigen Journale, genauso von der Qualität der hartumkämpften Turnierpartien wie von der präzisen Spielanalyse.

Jeder Schachspieler wird sich wohl schon einmal mit der Frage befaßt haben, welchen Stil die häusliche Partienanalyse zu befolgen hätte, um möglichst effektiv verzweigte Variantensysteme in allen Phasen der Partie durchzurechnen. Die Grenze lag dann meistens nicht in der Verläßlichkeit des Gedächtnisses, sondern im Erlöschen der Konzentration. Dabei verlor man das meiste an Zeit mit dem Auf- und Abbauen von Positionen. Wie sehr hat man sich dann ein Hilfsmittel herbeigewünscht, daß sich Grundstellungen von Positionen merken könnte? NICBASE trägt diesem Schachspielerwunsch Rechnung.

Eine Partie läßt sich bis zu einer theoretisch interessanten Phase nachspielen, dann in den Analyse-Modus schalten und variantenweise untersuchen. Pro Halbzug steht die Möglichkeit bereit, bis zu fünf Zugalternativen anzugeben. Durch Anklicken des Figurensymbols mit der Maus kann die Analyse auf dem Schachbrett mitvollzogen werden. Hierin zeigt NICBASE seine überragende Funktion, wichtige Abweichungen von der Brettstellung systematisch zu erfassen. Die Analyse wird alternativ mit der Maus, wenn Züge ausgeführt, und mit den Cursor-Tasten (rechter und linker Pfeil) gesteuert, wenn Zugfolgen rekonstruiert werden.

NICBASE bietet eine Reihe der üblichen Schachsonderzeichen an, wie sie aus der Literatur vertraut sind. Aufgerufen werden sie über ALTERNATE-Taste. Leider findet sich nirgends im Manual ein Bedeutungsindex, so daß man bei verschiedenen Zeichen herumrätseln kann, welches Bewertungskriterium sie signalisieren. Darüber hilft auch nicht die Tatsache hinweg, daß das Schachdatenbanksystem für Professionals entworfen ist, denn auch in der Schachwelt unterhält man sich über Konventionen. Sogar manuell eingegebene Varianten, die einen Kommentar zu einer Position darstellen, können über den Cursor-Block nachgespielt werden. Nur anfangs muß man sich ein wenig in Konzentration üben, denn die Bedienung der Analysefunktion erfordert eine genaue Reihenfolge der Befehle, wenn nicht die eine oder andere Variante unbesehen das RAM wieder verlassen soll. Der Analysemodus blockiert die Datenbank bis zum Abschluß für weitere Arbeitsgänge. Nach Beendigung der Analyse muß alles abgespeichert werden.

NICBASE bietet drei verschiedene Speicheroptionen an. Einmal kann der einzelne Partiedatensatz durch den durch Analyse und Kommentar ergänzten Partietext überschrieben werden. Dann ist es möglich, ihn als sog. Gamefile abzuspeichern. Das sind Partien, die keine Klassifizierung brauchen und trotzdem eingelesen werden können. Und schließlich mit dem Extender .PAR, mit dem ein bearbeiteter Datensatz üblicherweise zu sichern ist. Dank der verschiedenen Extender-Kennzeichnungen ist ein erstes grobes Durchgliedern vieler Datensätze, also Partien, kein Problem. Viele Schachspieler werden sicherlich mit mir übereinstimmen, wenn ich sage, daß mit einem solch komfortablen Analysemodus dem geheimnisvollen Dschungel Mittelspiel, wirkungsvoll den versteckten Gefahren trotzend, die Machete gezeigt werden kann. Es wird aber noch besser. Denn alle ausgeheckten Varianten können im kombinierten Grafik-alphanumerischen Notationssystem mit ausführlichen Analysetexten kommentiert werden.

In der Regel gelten Schachspieler, wenn sie kein Fernschach betreiben, als schreibfaul. Bei dem gebotenen Analysekomfort muß aber schon ein Zweizehenfaultier bei der Geburt Pate gestanden haben, wenn die Qualitäten dieser speziellen Textverarbeitung nicht genutzt würden, um die ein oder andere Zugfolge (wenn auch mit der Kompetenz der Wohnzimmeratmosphäre) nach allen Regeln der Kunst genüßlich zu kommentieren. Gerade auch der Profi bekommt ein Werkzeug an die Hand, das ihm wie willkommen erscheinen muß, detaillierte Analysen am Bildschirm zu erstellen und, da das eine das andere mehrt, zu verwalten. Mit den vielen Programmoptionen,

die gar nicht alle beschrieben werden können, präsentiert sich NICBASE als Schacharchiv mit Spitzenleistung, mit dem erst dann umfassend sinnvoll gearbeitet werden kann, wenn es als Schachdokumentation bei der täglichen Arbeit eingesetzt wird.

## Schach und Papier

Zudem stehen mehrere Druckoptionen zur Auswahl, für den aktiven Schachspieler sind insbesondere der Listendruck ganzer Schachpartientexte sowie der Ausdruck einzelner Positionen attraktiv. Denker, die sich mehr in spanischen Dörfern tummeln und Problemschach betreiben, können ihre Diagrammstempel auf dem Flohmarkt verhökern. Von nun an übernimmt NICBASE einen sauberen Diagrammdruck. Statistikern von Schachpartien steht eine Reihe einschlägiger Anzeigen zur Seite, die über die Anzahl von Gewinn-, Verlust-und Remispartien in einem Turnier aufklären oder sondieren, wievielmal Weiß mit einer Variante Erfolg auf sein Konto verbuchen konnte.

Sogar für die Spitzenkönner mit einem Gedächtnis wie eine Magnetnadel bietet NICBASE höchsten Komfort, denn es besteht die Möglichkeit, nach bestimmten Positionen und Stellungsmustern in der Schachdatenbank zu fahnden oder sich mit einem raffinierten Suchmanöver sämtliche Turmendspiele (ohne Beteiligung von Springer und Läufer) ausgeben zu lassen. Gerade durch die angebotene Quiz-Funktion wird deutlich, daß NICBASE ein System der hohen Leistungsklasse ist. Das Schachtraining mit dem Klavierspielen zu vergleichen, ist kein weither geholter Gedanke. Wie dort die Präzision der Tonfolge die Melodie macht, so müssen Zugfolgen konsequent aufeinander aufbauen, wobei den Spielern weite Freiräume für die Improvisation belassen sind.

Die Quiz-Funktion macht aus der Schachpartie ein Frage- und Antwortspiel. Das Spiel lebt von der Entscheidungsfreudigkeit des Spielers zwischen alternativen Zügen, die von Stellung zu Stellung wechseln, zu wählen. Solche Spielalternativen lassen sich nach einem Bewertungsschlüssel beurteilen. Das geschieht mit derselben Methode, wie eine scharfsinnige Analyse zu geschehen hat. Was in der Partienanalyse mit Intention und Verstand theoretisch <ausgekocht> wird, ist im Quiz einfach umgekehrt worden.

Der Analysator nimmt nun auf der Testbank Platz und gibt sein Urteil zu verschiedenen Partiensituation ab, die ihm vorgeführt werden. Für den Grad der Abweichung vom Textzug in der Quizpartie werden Punkte vergeben. Darüber, warum der Textzug und nicht ein alternativer in der Partiestellung zu geschehen hat, belehrt der beistehende Kommentar. NICBASE bietet immense didaktische Kapazitäten, das eigene Schachdenken zu trainieren und mit Fleiß und Ausdauer zu verbessern. Mit der Zeit könnte ein selbständig entwickeltes Trainingslehrbuch entstehen, das hilft, so manchen logischen Fehler transparent zu machen. Da NICBASE im Programmpaket ein solches Lernmedium per Quiz leider nicht anbietet, muß die Arbeit vom Anwender selbst geleistet werden.

Sehr auch kommt der Arbeit mit NICBASE entgegen, daß das System in modulare Teilprogramme aufgeteilt ist. Nicht zum Lieferumfang von NICBASE gehören vier Programme, die zwar zur internen Steuerung der Datenbank nicht benötigt werden, die aber für die Datenpflege unentbehrliche Dienste leisten.

Zum Aufpreis von hfl. 25.- /st. werden ein speziell für den Diagrammdruck mit Matrixdruckern ausgelegtes Druckprogramm NICPRINT, ein sehr nützliches Programm NICTWICE, das nach Partieduplikaten in Datenbanken sucht, ein Programm NICCLEAN, das eine Datenbank in der Urversion als Kopie herstellt, und als letztes ein Programm NICRECOV, das, bevor eine Reorganisation geschehen ist, gelöschte Partien reaktiviert, angeboten. Alle vier genannten Programme sind zusammen zum Sonderpreis von hfl 75.- zu erwerben.

## Kommunikation

Weil es immer besser ist, wenn unterschiedliche Systeme miteinander kommunizieren anstatt konkurrieren, wird vom INTERCHESS-Verlag für hfl 40.- das besondere Utility NICCONV angeboten. Es stellt den Dialog mit dem alternativen Schachdatenbanksystem CHESSBASE (V1.0, V2.0) her. Mit dieser Software können CHESSBASE-Partiesammlungen problemlos an das NICBASE-Notationssystem angepaßt und in eine bestehende Datenbank importiert werden. Für die Überspielung der Datensätze sollte man aber den Faktor Zeit einkalkulieren, mit einer Festplatte aber klappte der Transfer vorzüglich. Allerdings sind die überspielten Datensätze nicht nach dem spezifischen NICBASE-Eröffnungsschlüssel klassifiziert und müssen optional angepaßt werden. Die überspielten Daten werden als Gamefile erfaßt, können aber en gros klassifiziert werden. NICBASE wirbt damit, daß es 1800 Anpassungen/Std. vornimmt.

Zum Schluß sei noch berichtet, daß NICBASE mit Ausnahme gelegentlicher Assembler-Unterstützung bei kritischen GEM-Routinen in Turbo C entwickelt wurde. Der Arbeitscharakter von NICBASE wird daran am deutlichsten, daß eine Farbversion wegen der komplexen Dichte an darzustellender Information ausgeschlossen ist. Daß hier eine akkurate und speicherplatzsparende Programmierung zum Tragen kommt, muß nicht eigens betont werden. Pro Halbzug werden sechs Bits benötigt. Für eine Figurenposition werden höchstens 24 Bytes veranschlagt. Das Arbeitstempo ist für meinen Geschmack beim häufigen Umschalten zwischen Programmteilen ein wenig zu langsam. Das fällt besonders bei dem Analyseinventar auf. Ansonsten aber, insbesondere beim Rekonstruieren von Partien per rechtem Cursor-Pfeil, folgen die Halbzüge in Windeseile aufeinander. Mit dem Cursor-Pfeil unten sind sogar ganze Zugfolgen zurückzunehmen und so Ausgangsstellungen per Tastendruck wiederherzustellen. NICBASE, die intelligente Schachdatenbank aus Holland, der man den Arbeitscharakter sofort anmerkt, kostet in der Normalkonfiguration DM 228,-. Dazu müssen noch einige Zusatzprogramme erworben werden, für welche man sich nicht scheuen sollte, die Gulden über die Theke zu schieben. Und schließlich fallen laufende Kosten für auf Diskette archivierte NEW-IN-CHESS-Partieneditionen an, die mit der Zeit so manches Sümmchen verschlingen dürften. Daß nicht alles, was aus den Niederlanden kommt, Käse ist, haben wir kürzlich durch den Chemikalienfrachter Oostzee erfahren müssen. Nun kommt eine Schachdatenbank, die hat aber Zündstoff geladen, um der Intelligenz des Schachspielers kräftig einzuheizen.

*Ralf Blittkowsky*

***Bezugsadresse:***

*INTERCHESS BV*
*P.O. Box 393*
*NL-1800 AJ Alkmaar*
*Niederlande*

# Programmier Praxis

# Fast-Dfree

*Markus Fritze*

*JEDER KENNT BEI HARDDISKS DAS PROBLEM, DEN FREIEN SPEICHERPLATZ ZU ERMITTELN. DAS GEMDOS STELLT ZWAR EINE FUNKTION DAFÜR ZUR VERFÜGUNG, ABER BEI EINER SH205 MIT 4 PARTITIONEN DAUERT DAS FÜR ALLE ZUSAMMEN SCHON MAL 20 SEKUNDEN. MIT DIESEM KLEINEN PROGRAMM IM AUTO-ORDNER, DAS IM SPEICHER NICHT MAL 500 BYTES BELEGT, IST DIE DFREE-FUNKTION DES GEMDOS 14 (!!!)MAL SCHNELLER ALS VORHER, D.H. DIE GESAMTEN FREIEN CLUSTER DER HARDDISK HAT MAN IN CA. 1,4 SEKUNDEN!*

Diese kleine Routine ersetzt die *Dfree*-Funktion des GEMDOS vollständig. Nur bei Disketten (12-Bit-FAT) wird die (dort hinreichend schnelle) Originalroutine genutzt. Man kann die Routine auch sehr schön in eine Command-Shell implementieren, welche (wenn an IBM angelehnt) bei jedem *DIR* die freie Kapazität der Datenträgers ausgibt. Auch selbstprogrammierte File-Selektoren mit Laufwerksinfo profitieren von dieser Routine.

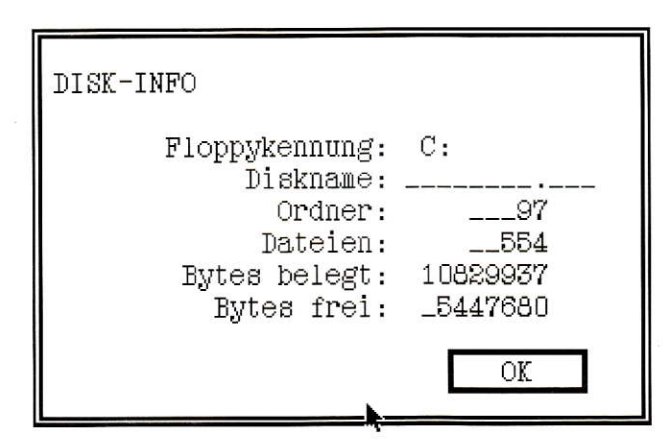
Besonderer Dank gilt an dieser Stelle Thomas Quester, der mich erst auf die Idee gebracht hat, eine solche Routine zu schreiben.

Das Programm ist zwar dokumentiert, aber hier noch einmal eine grobe Beschreibung der Vorgehensweise von *Fast-Dfree()*:

1. Bios-Parameter-Block des entsprechenden Laufwerks anfordern

2. bei 12-Bit-FAT in die Originalroutine ->

3. sonst die Gesamt-Cluster-Anzahl merken sowie den Startsektor der 2.FAT




4. 2 Sektoren der FAT einlesen (der Buffer ist gerade 1k groß).

5. Beim ersten Aufruf werden die ersten 3 Cluster nicht mitgezählt, da Sie vom GEMDOS aus nur intern verwandt werden.

6. Alle Words, welche gleich 0 sind, werden gezählt (16-Bit-FAT!).

7. Dabei wird die Gesamt-Cluster-Anzahl heruntergezählt,

8. wenn 512 Cluster gezählt wurden und die Gesamt-Cluster-Anzahl>0 ist, nach 4

9. Das war's.

Da das Programm mit dem OMIKRON.Assembler (Version 1.52) erstellt wurde, ist es gut möglich, daß andere Assembler z.B. den Pseudo-Opcode *OUTPUT* nicht kennen. Diesen kann man einfach weglassen. Ebenso ist mit der Adressierungsart Absolut-Short zu verfahren, welche beim OMIKRON.Assembler durch ein *.w* hinter dem Operator erzwungen werden kann.



```
 1:  ;===============================================
 2:  ;= Fast-Dfree() - beschleunigt die Dfree-      =
     = Funktion des Gemdos bei 16-Bit-FATS          =
 3:  ;= bei einer SH205 um Faktor 14 !!!            =
 4:  ;= Auf dem Desktop wird das Info 3mal schneller =
 5:  ;= Die neue Dfree-Funktion wird ins GEMDOS     =
     = eingesetzt und ersetzt bei                   =
 6:  ;= 16-Bit-FATS (Harddisk, Ramdisk, etc.) die   =
     = originale Dfree-Routine voll-                =
 7:  ;= ständig!                                    =
 8:  ;=                                             =
 9:  ;= Nach einer Idee von Thomas Quester (Danke)  =
10:  ;=                                             =
11:  ;= Enwtwickelt von Markus Fritze               =
12:  ;= (c) MAXON Computer GmbH 1990                =
13:  ;===============================================
14:                  OUTPUT 'DFREE'  ;bei andernen
                      Assemblern evtl.weglassen
15:
16:  anfang:         bra      init
17:
18:  ;===============================================
19:  ;= Der neue Gemdos-Vektor                      =
20:  ;===============================================
21:
```

→

```
22:                    DC.B 'XBRA'      ;XBRA-Standard
                                         f.vektorverbgnde
                                         Programme
23:                    DC.B '∑DFR'      ;Kennung von ∑-
                                         soft, Unterken-
                                         nung: DFREE
24:  old_vektor:       DS.L 1
25:  new_gemdos:       move      USP,A0
26:                    btst      #5,(SP) ;Aufruf aus dem
                                         User-Mode?
27:                    beq.s     new_gemdos1    ;Ja!
28:                    movea.l SP,A0
29:                    addq.l    #6,A0
30:
31:  new_gemdos1:      cmpi.w    #$36,(A0)      ;Dfree()
32:                    beq.s     dfree    ;eigene
                                          Routine testen
33:
34:  new_gemdos_exit:movea.l old_vektor(PC),A0
35:                    jmp       (A0)     ;Originale GEM-
                                          DOS-Routine
36:
37:  ;=================================================
38:  ;= Bei einer 12-Bit-FAT (Disketten) ist die      =
       = interne Routine schnell genug.              =
39:  ;= Deswegen haben ich mir die Arbeit gespart.    =
40:  ;=================================================
41:
42:  dfree_12bit_fat:movem.l (SP)+,D1-A6
43:                    bra.s     new_gemdos_exit
                                 ;Originalroutine aufrufen
44:
45:  ;=================================================
46:  ;= Die neue Dfree()-Funktion für 16-Bit-FATs     =
47:  ;= Es werden einfach alle Null-Words in der      =
       = 2.FAT gezählt, das ist alles.               =
48:  ;= Zu beachten ist lediglich, daP die ersten 3   =
       = Words nicht mitgezählt werden!              =
49:  ;=================================================
50:
51:  dfree:             movem.l D1-A6,-(SP)
52:                     movea.l 2(A0),A4
                         ;Bufferadr
53:                     lea       12(A4),A4;für
                                    Predecrement (s.u.)
54:
55:                     move.w    6(A0),D7 ;Drive
56:                     subq.w    #1,D7
57:                     bpl.s     dfree3    ;aktuelles LW?
58:
59:                     move.w    #$19,-(SP)
60:                     trap      #1        ;Dgetdrv()
61:                     addq.l    #2,SP
62:                     move.w    D0,D7     ;aktuelles LW
63:
64:  dfree3:            cmp.w     #1,D7     ;Disk?
65:                     ble.s     dfree_12bit_fat ;dann
                                           GEMDOS-Dfree()
66:  ;Wenn obige Abfrage fehlt, wird nach dem
      Formatieren einer einseitigen Disk
67:  ;von eine 16-Bit-FAT ausgegangen (fehlendes
      Media-Change)
68:
69:                     move.w  D7,-(SP) ;drive
70:                     move.w  #7,-(SP)
71:                     trap      #13       ;Getbpb (drive)
72:                     addq.l  #4,SP
73:                     tst.l     D0
74:                     bmi.s     dfree_error ;Gerät nicht
                                           da, Fehler
75:                     movea.l D0,A6       ;Adresse merken
76:
77:                     btst      #0,17(A6);Flagsb testen
                          nach 16 oder 12-Bit-FAT
78:                     beq.s    dfree_12bit_fat ;12-Bit-
                         FAT =>Original ist schnell genug
79:
80:                     move.w  10(A6),D6 ;fatrec -
                                 Startsektor der 2.FAT
81:
82:                     moveq     #0,D5
83:                     move.w  2(A6),D5
84:                     move.l   D5,(A4)      ;Sektoren pro
                                            Cluster einsetzen
85:
86:                     move.w  (A6),D5
87:                     move.l  D5,-(A4);Bytes pro Sektor
                                        einsetzen
88:
89:                     move.w  14(A6),D5 ;numcl -
                         Gesamtanzahl der Cluster merken
90:                     move.l  D5,-(A4)   ;und einsetzen
91:
92:                     moveq     #0,D4      ;Anzahl
                              der freien Cluster=0
93:                     moveq     #0,D3      ;Flag für den
                          Startsektor der FAT löschen
94:  dfree0:            movea.l $04C6.w,A5
                       ;Bufferadresse holen (_dskbuf)
95:                     move.w  D7,-(SP) ;Drive
96:                     move.w  D6,-(SP) ;fatrec
97:                     move.w  #2,-(SP) ;2 Sektoren
                                          einlesen
98:                     move.l  A5,-(SP) ;Buffer für die
                                          Sektoren
99:                     clr.w     -(SP)     ;normales Lesen
100:                    move.w  #4,-(SP)
101:                    trap      #13        ;Rwabs()
102:                    lea       14(SP),SP
103:                    tst.l     D0
104:                    bmi.s     dfree_error
               ;Lesefehler, Fehlermeldung zurückgeben
105:
106:                    addq.w  #2,D6      ;fatrec+2
107:
108:                    move.w  #$01FF,D0;512 Cluster pro
                                      2 Sektoren der FAT
109:
110:                    tas.b     D3
             ;1.Sektor mit den ersten drei Clustern?
111:                    bne.s     dfree1     ;Nein! =>
112:
113:                    addq.l  #6,A5       ;die ersten der
                                          Cluster werden
114:                    subq.w  #3,D0   ;nicht mitgezählt!
115:                    subq.w  #3,D5       ;3 Cluster
                                      bereits abziehen
116:
117:  dfree1:           tst.w     (A5)+       ;freien
                                    CLuster gefunden?
118:                    bne.s     dfree2     ;Nein! =>
119:                    addq.w  #1,D4       ;einen freien
                                      Cluster gefunden
120:  dfree2:           subq.w  #1,D5       ;numcl-1
121:                    dbeq      D0,dfree1
122:                    bne.s     dfree0     ;Ende noch nicht
                                  erreicht, weiter geht's
123:
124:                    move.l  D4,-(A4)  ;Anzahl der
                              freien Cluster einsetzen
125:                    moveq     #0,D0      ;alles ok, kein
                                           Fehler
126:
127:  dfree_error:      movem.l (SP)+,D1-A6
128:                    rte                  ;Das war's schon
129:
130:  init:             pea       new_gemdos(PC)
131:                    move.l  #$050021,-(SP)
132:                    trap      #13              ;Setexc
                                         (33,new_gemdos)
133:                    addq.l  #8,SP
134:                    move.l  D0,old_vektor  ;Original-
                                          Vektor merken
135:
136:                    pea       init_text(PC)
137:                    move.w  #9,-(SP)
138:                    trap      #1           ;eine Meldung am
                                           Anfang
139:                    addq.l  #6,SP
140:
141:                    clr.w     -(SP)        ;Kein Fehler
                                          aufgetreten
142:                    pea       init-anfang+$0100.w
                      ;residentes Programm <500 Byte
143:                    move.w  #$31,-(SP)
144:                    trap      #1           ;Ptermres()
145:
146:                    DATA
147:  init_text:        DC.B 'Fast-Dfree',13,10
148:                    DC.B '(c) 1990 MAXON Computer
                          GmbH, von Markus Fritze'
                          ,13,10,0
149:                    END
```

# Bilddaten-Konverter

*Hardy Illsinger*

Das hier vorgestellte Programm ist in der Lage, Bilder als Ganzes oder nur Bildausschnitte in Datazeilen umzuwandeln. Dadurch hat man die Möglichkeit, kleine Grafiken in eigene Programme einzubauen, ohne vorher noch ein Bild laden zu müssen.

Das Herz des Programms ist der Befehl *POINT (X,Y)*, mit dem abgefragt wird, ob ein Bildpunkt (Pixel) vorliegt oder nicht. Mit der FOR-NEXT-Schleife wird dann der gewünschte Bildausschnitt durchlaufen und nach obigem Prinzip abgefragt. Hat das Programm einen Punkt gefunden, schreibt es sofort eine DATA-Zeile auf Diskette. In einer DATA-Zeile befinden sich max. neun Koordinaten mit den x- und y-Werten. Werden noch mehr Koordinaten gefunden, eröffnet das Programm eine neue DATA-Zeile auf Diskette. Ferner wird noch eine kleine Routine mit abgespeichert, mit der die abgespeicherten DATA- Zeilen sofort als Programm gestartet werden können.

## Einladen und fertig

Natürlich muß man das abgespeicherte Programm vorher erst mit MERGE im Interpreter laden, sonst tut sich gar nichts. Lädt man das Programm *DATA.LST* dann in den Interpreter, braucht man es nur noch zu starten, und man bekommt die vorher eingegrenzten Grafikzeilen. Wie man die Bildpunkte vorher eingrenzt? Nun, das ist eigentlich recht einfach.

Zunächst tippt man das Programm *BILDDATA.BAS* ab (möglichst fehlerfrei), und wenn man will und kann, hat man sogar die Möglichkeit, es zu compilieren. Der nächste Schritt ist dann der, das Programm zu starten. Anschließend erscheint eine Fileselectorbox, mit der Sie das gewünschte Bild auswählen sollen, aus der Sie die Grafik entnehmen wollen. Ist auch dies geschehen, wird das Bild geladen. Jetzt kommt der eigentliche Akt: Mit einem Fadenkreuz ziehen Sie bei gedrückter linker Maustaste eine Box um den gewählten Bereich. Das Programm beginnt dann mit der schon oben beschriebenen Arbeit. Nach Beendigung wird es sofort verlassen.

Das war's. Im Moment ist das Programm nur in der Lage, SCREEN-Bildformate (Doodle) zu laden, aber das kann natürlich geändert oder erweitert werden, ganz wie man möchte. Weiterhin möchte ich noch anmerken, daß der gewählte Bildinhalt nicht zu groß sein sollte, da sonst der Rechner später seine Zeit damit vertrödelt, mit der ohnehin sehr langsamen MERGE-Laderoutine Unmengen von Daten zu laden. Wenn dann der RAM-Speicher auch noch knapp wird, hat man vielleicht umsonst eine Kaffeepause eingelegt.



Nun bleibt mir nur noch, Ihnen viel Erfolg bei der Gestaltung Ihrer Programme mit hübschen Grafiken zu wünschen.

```
 1:  ' Bildausschnitte umwandeln in Datazeilen
 2:  '
 3:  ' Von Hardy Illsinger, Birkenweg 16, 6110 Dieburg
 4:  '
 5:  ' (c) MAXON Computer GmbH 1990
 6:  '
 7:  CLS
 8:  SETCOLOR 0,1
 9:  BOX 149,10,490,52
10:  DEFTEXT 1,0,0,13
11:  TEXT 156,32,"Bitte wählen Sie das zu bearbeitende
                Bild"
12:  DEFTEXT 1,0,0,6
13:  TEXT 360,46,"Doodle-Format"
14:  FILESELECT "\*.PIC","",bild$
15:  IF bild$=""
16:    CLS
17:    ALERT 1,"ABBRUCH gedrückt !!!",2,"weiter|ENDE",
           wahl%
18:    IF wahl%=1
19:      CLS
20:      RUN
21:    ENDIF
22:    IF wahl%=2
23:      CLS
24:      EDIT
25:    ENDIF
26:  ENDIF
27:  CLS
28:  BLOAD bild$,XBIOS(2)
29:  '
30:  DEFMOUSE 5  !Routine zur Wahl des Bildausschnitts
31:  DO
32:    MOUSE xl%,yl%,k%
33:    GRAPHMODE 3
34:    DEFLINE 2,1,0,0
35:    WHILE MOUSEK=1
36:      MOUSE xr%,yr%,k%
37:      BOX xl%,yl%,xr%,yr%
38:      REPEAT
39:      UNTIL MOUSEK=0 OR MOUSEX<>xr% OR MOUSEY<>yr%
40:      BOX xl%,yl%,xr%,yr%
41:      IF MOUSEK=0
42:        @data
43:      ENDIF
44:    WEND
45:  LOOP
46:  '
47:  PROCEDURE data          ! Routine zur Abtastung
                               der Bildpunkte
48:    OPEN "O",#1,"DATA.LST"
49:    PRINT #1,"@Zaehlauswertung"
50:    PRINT #1,"For I=1 To Zaehler"
51:    PRINT #1,"Read X,Y"
52:    PRINT #1,"Plot X,Y"
53:    PRINT #1,"Next I"
54:    PRINT #1,"Void Inp(2)"
55:    zeile=0
56:    zaehler=0
57:    FOR y=yl% TO yr%
58:      FOR x=xl% TO xr%
59:        IF POINT(x,y)=1
60:          INC zaehler
61:          IF zeile=0
62:            PRINT #1,"Data ";
63:          ENDIF
64:          PRINT #1,x;",";y;
65:          INC zeile
66:          IF zeile<9
67:            PRINT #1,",";
68:          ENDIF
69:          IF zeile=9
70:            PRINT #1,""
71:            zeile=0
72:          ENDIF
73:        ENDIF
74:        IF y=yr% AND x=xr%
75:          PRINT #1,""
76:        ENDIF
77:      NEXT x
78:    NEXT y
79:    PRINT #1,"Procedure Zaehlauswertung"
80:    PRINT #1,"Zaehler=",zaehler
81:    PRINT #1,"Return"
82:    CLOSE #1
83:    EDIT
84:  RETURN
```

# Turbo C überlistet

*Jan Bolt*

*TURBO C VON BORLAND BIETET EINE RECHT ANSPRECHENDE BENUTZEROBERFLÄCHE. MÖCHTE MAN JEDOCH DEN COMPILER IN EINER EIGENEN SHELL ODER EINEM BATCH-MONITOR INSTALLIEREN, STÖSST MAN AUF ZWEI MÄNGEL DES PROGRAMMS TCC.TTP.*

Erstens erscheinen alle Fehlermeldungen und Warnungen auf dem Bildschirm. Mehr als drei Meldungen kann man sich kaum merken, wenn man wieder den Editor aufruft.

Zweitens liefert der Compiler an das aufrufende Programm nur bei Fehlermeldungen einen Wert ungleich Null zurück, nicht jedoch bei Warnungen.

Abhilfe schafft das folgende Programm *TURBO.TTP*, das anstelle von *TCC.TTP* aufgerufen wird. *TURBO.TTP* wird die gleiche Kommandozeile übergeben, mit der man *TCC.TTP* aufgerufen hätte. *TURBO.TTP* leistet nun folgendes: Die gewohnte Copyright-Meldung wird ausgegeben und anschließend die Bildschirmausgabe mit den GEMDOS-Funktionen Fdup und Fforce in die Datei *ERRORS.OUT* umgeleitet. Dann ruft man den Turbo C-Compiler *TCC.TTP* mittels Pexec auf. Dabei werden die Kommandozeile und der *environment pointer* von *TURBO.TTP* an *TCC.TTP* weitergegeben. Nachdem der Compiler seine Arbeit vollendet hat, wird die Bildschirmausgabe wieder mit dem Bildschirm verbunden und die Datei *ERRORS.OUT* ausgegeben, falls sie Meldungen enthält. -1 wird an das aufrufende Programm zurückgegeben. Ist die Datei leer, gibt *TURBO.TTP* 0 zurück. Mit einem Editor, der zwei Textfenster bearbeiten kann (z.B. TEMPUS), läßt sich nun der Quelltext korrigieren, während die Fehlermeldungen im zweiten Fenster gehalten werden.



```
 1:  /*============================================*
 2:   * TURBO.C                                    *
 3:   *                                            *
 4:   * Umlenkung der Bildschirmausgabe von TURBO C *
 5:   * in Datei "ERRORS.OUT"                      *
 6:   *   - Umlenkung der Bildschirmausgabe in Datei*
 7:   *   - Nachladen und Starten von "TCC.TTP"    *
 8:   *   - alle Bildschirmausgaben von "TCC.TTP"
          gehen in Datei "ERRORS.OUT"              *
 9:   *   - Kommandozeile und environment pointer  *
10:   *     werden an "TCC.TTP" weitergegeben      *
11:   *   - Bildschirmausgabe zurück auf Bildschirm *
12:   *   - Ausgabe der Datei "ERRORS.OUT"         *
13:   *   - Rückgabe an das aufrufende Programm :  *
14:   *       0 : keine Meldungen von TCC.TTP      *
15:   *      -1 : mindestens 1 Warnung/Fehlermeldung *
16:   * 20.10.89  Jan Bolt                         *
17:   * (c) MAXON Computer GmbH 1990               *
18:   * TURBO C  V 1.0                             *
19:   *============================================*/
20:
21:  /* Linken mit TLINK Option -S=512
22:  (512 bytes stack reichen) */
23:
24:  #include <stddef.h>
25:  #include <stdio.h>
26:  #include <tos.h>
27:  #include <string.h>
28:
29:  #define CON 1
30:  #define OUTPUT "ERRORS.OUT"
31:  #define OVERLAY "TCC.TTP"
32:
33:  extern BASPAG *_BasPag;          /* Zeiger auf
                              Basepage aus TCSTART */
34:
35:  char copyright [] = "\x0d\x0a\
36:  Turbo C   Version 1.00  Copyright (c) 1988 \
37:  Borland International\x0d\x0a";
38:
```

→

```
39:  int type (char *fname)          /* Datei auf
                                 Bildschirm ausgeben */
40:       {
41:       char buffer [80];
42:       int fh, k, i;
43:       int test = 0;
44:
45:       if ((fh=Fopen(fname,0)) >= 0)
46:            {
47:            Fseek ((long)strlen(copyright),fh,0);
48:            if ((test=(int)Fread(fh,11,buffer)) == 1)
49:                 Cconout (buffer[0]);
50:            while ((k=(int)Fread(fh,801,buffer)) > 0)
51:                      {
52:                      i = 0;
53:                      while (i <= k-1)
54:                                Cconout ((int)buffer[i++]);
55:                      }
56:            }
57:       if (test != 0)
58:            test = -1;
59:       return (test);
```

```
60:       }
61:
62:  int main (int argc,char *argv[],char *env)
63:       {
64:       int out, con2, tyret;
65:
66:       Cconws (copyright);
67:       if ( ((out = Fcreate (OUTPUT,0)) >= 0) &&
68:               ((con2 = Fdup (CON)) >= 0) )
69:            {
70:            Fforce (CON,out);
71:            Pexec (0,OVERLAY,(COMMAND *)_BasPag-
                 >p_cmdlin,env);
72:            Fforce (CON,con2);
73:            Fclose (out);
74:            tyret = type (OUTPUT);
75:            }
76:       else
77:            Cconws ("I\O-error !\x0d\x0a");
78:       return (tyret);
79:       }
```

# Commodore 1581-Wandler

**Helmut Büche**

*DAS 3,5"-LAUFWERK 1581 VON COMMODORE FÜR DIE RECHNER C64 UND C128 VERWENDET DAS GLEICHE PHYSIKALISCHE AUFZEICHNUNGSFORMAT WIE DAS LAUFWERK DES ATARI ST, NÄMLICH DAS MFM-FORMAT. DAMIT ERGIBT SICH FÜR UMSTEIGER VOM COMMODORE 64/128 ZUM ATARI ST EINE EINFACHE MÖGLICHKEIT FÜR DEN DATEITRANSFER.*

Zwar ist das File-Management auf Commodore-Disketten gegenüber dem auf ATARI-Disketten völlig verschieden, aber beide Rechner können die Disketten des anderen Rechners im Direktzugriff auf Spur und Sektor lesen. Dieser Sachverhalt liegt dem vorliegenden Programm zugrunde.

## Falsche Seite

Mittels des XBIOS-Befehls *XBIOS(r,8...)* läßt sich eine Spur einer Commodore-Disk einlesen. Sie umfaßt 20 Commodore-Sektoren zu je 256 Bytes, was 10 ATARI-Sektoren entspricht. Da das Commodore-Laufwerk generell beidseitig aufzeichnet, verdoppelt sich die Sektorenanzahl pro Spur. Interessant ist, daß bei Commodore die Seitenbezeichnung gegenüber dem ATARI vertauscht ist. So liegt der erste Sektor einer Spur generell auf Seite 1 und der letzte Sektor auf Seite 0.

Mit dem Programm lassen sich Commodore-Files vom Typ *PRG, SEQ* und *USR* in das ATARI-Format konvertieren. Andere File-Typen wie relative Files *(REL)* und Ordner *(CBM)* werden von diesem Programm nicht unterstützt. Wer eine Konvertierung solcher Dateien will, kann das Programm leicht selbst erweitern, da es in Omikron.BASIC geschrieben und somit einfach zu modifizieren ist. Allerdings ist dazu eine gute Kenntnis des File-Managements der Commodore-Disk notwendig.

```
READY.
?OUT OF DATA ERROR

READY.
L"$",8
SEARCHING FOR $
LOADING

READY.
L
00 "PECES WABBELDISK" 2A ST
65  "EXDOS&DISKDOC"     PRG
125 "JABBADEMO"         PRG
67  "ULTIMAANLEITUNG"   SEQ
68  "USERDATEN"         REL
33  "BITMAPBROTHERS"    SEQ
121 "VIZAWRITE"         PRG
185 BLOCKS FREE.

READY.
SYS 64738
```

Für die Konvertierung werden zwei Laufwerke am ATARI benötigt. In Laufwerk A wird die Commodore-Disk, in Laufwerk B eine formatierte ATARI-Disk eingelegt. Nach dem Start des Programms erfolgt eine Kurzinformation und die Abfrage, ob zunächst ein Testlauf, bei dem kein Schreibvorgang erfolgt, durchgeführt werden soll. Anschließend wird das Directory der Commodore-Disk eingelesen. In einer weiteren Abfrage kann man sich dafür entscheiden, ob generell alle Dateien vom Typ *SEQ*, oder ob nach einer Auswahl konvertiert werden soll. Dazu ist zu sagen, daß es im allgemeinen wegen der vollkommen unterschiedlichen Prozessoren und den anders aufgebauten BASIC-Interpretern nicht sinnvoll ist, Programm-Files zu konvertieren (PRG), es sei denn, daß in solchen Files Texte abgelegt sind, wie es z.B. das Textprogramm Vizawrite macht. In der Regel wird man Text-Files vom Typ SEQ konvertieren. Bei der pauschalen Umwandlung von *SEQ*-Dateien erfolgt generell eine Code-Umwandlung, da der Commodore-Rechner sich nicht sehr eng an den ASCII-Standard hält. Bei Konvertierung nach

Auswahl kann man bei jeder Datei durch Eingabe von "J" entscheiden, ob eine Code-Umwandlung erfolgen soll oder nicht. Bei einer anderen Eingabe als "J" wird die Datei nicht ausgewählt und ein "Nein" hinter den Dateinamen gehängt. Ist das Ende der File-Liste erreicht, erfolgt die Abfrage, ob die Auswahl wiederholt werden soll, um die getätigten Angaben eventuell noch einmal ändern zu können.

## Universelle Anpassung

Im Programm selbst ist die Routine zur Code-Wandlung mit dem Label *"-Codewdlg"* bezeichnet. Hier lassen sich zusätzliche, individuelle Code-Wandlungen einfügen, welche mit dem Flag *"Zus"* aktiviert werden. Während der Konvertierung werden der Dateiname und der Inhalt der Datei auf dem Bildschirm angezeigt. Weil die zulässige Dateinamenlänge beim ATARI ST gegenüber 16 Zeichen beim Commodore nur 8 Zeichen beträgt, wird jedem konvertierbaren File eine laufende Nummer zugewiesen. Die umgewandelten Dateien erhalten als Dateinamen diese Nummer. Nach dem Konvertieren der Dateien schreibt das Programm eine zusätzliche Informationsdatei auf die Diskette. In diesem File mit dem Namen *"Namelist.Dat"* sind sämtliche Namen der konvertierten Dateien unter der laufenden Nummer enthalten. PP

```
 1:   '  V 1.1,  12.11.1989
 2:   '  (c) MAXON Computer GmbH 1990
 3:   '  *****************************************
 4:   '  *     Programm zur Konvertierung von    *
 5:   '  *       Files im COMMODORE-Format       *
 6:   '  *      des Diskettenlaufwerkes 1581     *
 7:   '  *     in Files des ATARI ST-Formates    *
 8:   '  *****************************************
 9:   '
10:      ON ERROR GOTO Fehler
11:      DEFINT "D-U"
12:      Z%L=512*20:Adr0%L= MEMORY(Z%L):Adr1%L=Adr0%L+
             Z%L/2
13:      DIM Feld%B(Z%L): DIM Name$(304)
14:      DATA "DEL","SEQ","PRG","USR","REL","CBM"
15:      FOR I%=0 TO 5: READ E$:Ftyp$(I%)=E$: NEXT
16:      Zus%L=1'   Flag für zus., individuelle Code-
                 Wandlungen
17:      PRINT CHR$(27);"H"; CHR$(27);"H";CHR$(27);"E"
18:   '
19:      E$= CHR$(34)+" COMMODORE-FILE -> ATARI-FILE "
           + CHR$(34)
20:      TEXT STYLE =8: TEXT 168,15,E$,300,0,1
21:      PRINT "    "; CHR$(4)
22:      PRINT " Das Programm konvertiert PRG-, SEQ-
             und USR-Files im Format des"
23:      PRINT " COMMODORE Laufwerkes 1581 in
             entsprechende Files des ATARI ST"
24:      PRINT " mit den Extensionen PGM, TXT und
             USR."
25:      PRINT " Files vom Typ REL und CBM können
             nicht konvertiert werden."
26:      PRINT
27:      PRINT " Es müssen zwei Laufwerke vorhanden
             sein und die Zieldiskette";
28:      PRINT " muß bereits": PRINT " formatiert
             sein !"
29:      PRINT
30:      PRINT " Testlauf (ohne schreiben) ?  ";:E$=
           INPUT$(1): PRINT CHR$(27);"M"
31:      IF E$="n" OR E$="N" THEN Wrt%L=1: ELSE PRINT
           TAB (27);">>   TESTLAUF   <<"
32:      PRINT CHR$(27);"B"
33:      PRINT TAB (20);" COMMODORE-Diskette in
                       Laufwerk A"; CHR$(27);"B"
34:      IF Wrt%L THEN PRINT TAB (20);" ATARI ST -
           Diskette in Laufwerk B"; CHR$(27);"B"
35:      PRINT TAB (27);"      einlegen !"
36:      PRINT @(20,27);" <Taste, wenn OK>   ";:E$=
           INPUT$(1)
37:      PRINT CHR$(27);"l"; CHR$(27);"f"; CHR$(13),,,
           "     bitte einen Moment ..."
38:   '
39:   '  Directory der 1581 einlesen:
40:      T%=40: GOSUB Track
41:           IF Ds% OR Feld%B(0)<>40 OR
                Feld%B(1)<>3 THEN
42:             PRINT @(20,16); CHR$(27);"J Im
                   Laufwerk A ist keine COMMODORE-
                   Disk !": END
43:           ENDIF
44:           PRINT CHR$(27);"E"; CHR$(27);"B
                 DIRECTORY:"; CHR$(27);"p";:E$=
                 CHR$(27)+"p"+ CHR$(34)
45:           FOR I%=4 TO 26
46:            IF I%=20 THEN E$=E$+ CHR$(34):I%=I%+1
47:             U%=Feld%B(I%): GOSUB Codewdlg: IF U%
                   THEN E$=E$+ CHR$(U%)
48:           NEXT :E$=E$+ CHR$(27)+"q":Name$(0)=E$
49:           PRINT TAB (25);E$: PRINT
50:           PRINT @(21,79); CHR$(8);@(3,0);CHR$(4);
51:           Of%=256*3+2'  File-Namen ab Sektor 3
                         (COMMODORE-Format)
52:      REPEAT
53:             J%=Of%:T%=Feld%B(J%): IF T%=0 THEN
                                     GOTO Nxt
54:             E$="":F%= VAL("$"+ RIGHT$( HEX$(T%),
                   1)):P%=16
55:            FOR I%=J% TO J%+2:E$=E$+
                 CHR$(Feld%B(I%)): NEXT
56:            FOR I%=J%+3 TO J%+19:U%=Feld%B(I%):
                 IF U%=160 THEN P%=I%-J%-3:EXIT
57:              GOSUB Codewdlg: IF U% THEN E$=E$+
                                          CHR$(U%)
58:            NEXT
59:           U%=Feld%B(J%+28)+Feld%B(J%+29)*256'
                Blockzahl
60:           H$= LEFT$(E$,3)+ LEFT$( STR$(U%)+"
                 ",5)+ CHR$(34)
61:           H$=H$+ LEFT$( MID$(E$,4,P%)+ CHR$(34)+
                SPC(20),18)+Ftyp$(F%)
62:           E$="*": IF T%>127 THEN E$=" ": IF
                T%>191 THEN E$="<"
63:           N%=N%+1:Name$(N%)=H$+E$: PRINT TAB
                (23); MID$(Name$(N%),4)
64:          -Nxt:Of%=J%+32
65:      UNTIL Feld%B(J%+3)=0
66:       PRINT :H$="Code-Wandlung"
67:       PRINT "konvert :    1 = alle SEQ-Files <mit
             ";H$;">,   2 = nach Auswahl     ";
             CHR$(27);"e";
68:       REPEAT :E$= INPUT$(1): UNTIL E$="1" OR
             E$="2"
69:       IF E$="1" THEN Alle%L=1: GOTO Konvert
70:   '
71:      -Ausw: PRINT CHR$(27);"H"; CHR$(27);"H";
             CHR$(27);"E"; CHR$(27);"B"; TAB
             (27);Name$(0): PRINT CHR$(27);"B";
             CHR$(4);@(21,0);
72:        I%=1: PRINT CHR$(27);"c"; CHR$(I%);
             CHR$(27);"J"; CHR$(27);"b";
             CHR$(I%+1); SPC(8);"ja = mit ";H$;
73:          PRINT "      JA = ohne ";H$;"       * =
                 Abbruch";
74:          PRINT CHR$(27);"b"; CHR$(I%);
                 CHR$(27);"c"; CHR$(I%+1);
75:          PRINT CHR$(27);"q";@(19,79); CHR$(8);
                 CHR$(27);"E";:-Auswl
76:       A$="  "
77:       FOR I%=1 TO N%
78:         N$=Name$(I%):H$= MID$(N$,32):U%=
              ASC(N$):H%= VAL("#"( RIGHT$( HEX$(U%),1))
79:         PRINT TAB (18);"#"; RIGHT$(A$+ STR$(I%),
               3);A$; MID$(N$,4,28);A$;
80:         IF U%<127 OR H%=0 OR H%>3 THEN PRINT :
             GOTO Ni
81:         IF H$<>"" THEN
82:           IF H$="j" THEN PRINT "ja      ";
83:           IF H$="J" THEN PRINT "JA      ";
84:           IF H$<>"j" AND H$<>"J" THEN PRINT "
               nein ";
85:         ELSE PRINT "ja/nein ";
86:         ENDIF
87:         E$= INPUT$(1): PRINT CHR$(13); TAB (56);
88:           IF E$="+" OR E$="*" THEN PRINT CHR$(27);
               "l"; CHR$(13);: EXIT
89:         IF H$="" THEN
```

→

```
 90:          IF E$<>"j" AND E$<>"J" THEN PRINT
              "   nein "
 91:        ELSE
 92:          IF E$<>"j" AND E$<>"J" AND E$<>"n" THEN
              PRINT : GOTO Ni
 93:          IF E$="n" THEN PRINT "    nein "
 94:        ENDIF
 95:          IF E$="j" THEN PRINT "ja      "
 96:          IF E$="J" THEN PRINT "JA      "
 97:        Name$(I%)= LEFT$(N$,31)+E$
 98:      -Ni: NEXT
 99:       PRINT CHR$(27);"f"; CHR$(27);"B";
             CHR$(27);"A";: REPEAT : WAIT .1:E$=
             INKEY$ : UNTIL E$=""
100:       INPUT CHR$(27)+"B Auswahl wiederholen ? ";
             CHR$(27);"e";E$ USING "+j+n s$s1",
             Cr%L,1,32
101:       IF RIGHT$( HEX$(Cr%L),1)="A" THEN PRINT
             CHR$(27);"E";: GOTO Auswl
102:
103:      -Konvert:T%=0:U%=0: CLS
104:        FOR I%=1 TO N%:H$= RIGHT$(Name$(I%),1): IF
             H$="j" OR H$="J" THEN U%=I%
105:        NEXT : IF Alle%L THEN U%=N%
106:        E$="B:\NAMELIST.DAT"
107:        IF Wrt%L*U% THEN OPEN "O",1,E$: PRINT #1,"
             "; MID$(Name$(0),3,24)
108:        FOR H%=1 TO U%
109:          E$=Name$(H%):Tn%= ASC( MID$(E$,2,1)):S%=
             ASC( MID$(E$,3,1))
110:          IF Alle%L=0 AND RIGHT$(E$,1)<>"j" AND
             RIGHT$(E$,1)<>"J" THEN GOTO Nha
111:          F%= VAL("$"+ RIGHT$( HEX$( ASC((E$))),1))
112:          IF Alle%L=1 AND F%<>1 THEN GOTO Nha
113:          N$= MID$(E$,4,28):H$=Ftyp$(F%)
114:          IF F%=1 THEN H$="TXT"
115:          IF F%=2 THEN H$="PGM"
116:          H$="#_"+ RIGHT$("000"+ MID$( STR$(H%),2),
             3)+"."+H$
117:          Name$(H%)=N$+"    ->    "+ CHR$(34)+H$+
                  CHR$(34)
118:          IF Wrt%L THEN OPEN "O",2,"B:\"+H$: PRINT
             #1,Name$(H%)
119:          PRINT CHR$(27);"H"; CHR$(27);"H";
                  CHR$(27);"E"
120:          PRINT TAB (16);CHR$(27);"p";N$;
                                CHR$(27);"q  ->  ";
                                CHR$(27);"p"; CHR$(34);H$;
                                CHR$(34);" "; CHR$(27);"q"
121:          PRINT CHR$(27);"B"; CHR$(4);
122:          REPEAT
123:            IF Tn%<>T% THEN T%=Tn%: GOSUB Track
124:            Of%=256*S%:P%=255:Tn%=Feld%B(Of%):
                S%=Feld%B(Of%+1): IF Tn%=0 THEN P%=S%
125:            FOR I%=2 TO P%
126:              U%=Feld%B(I%+Of%)
127:              IF Alle%L OR RIGHT$(E$,1)="j" THEN
                  GOSUB Codewdlg: IF U%=0 THEN GOTO Qq
128:              PRINT CHR$(1); CHR$(U%);
129:              IF Wrt%L THEN PRINT #2, CHR$(U%);: IF
                  U%=13 THEN IF F%=1 THEN PRINT
                  #2, CHR$(10);
130:            -Qq: NEXT
131:          UNTIL Tn%=0: CLOSE 2
132:      -Nha: NEXT : CLOSE 1
133:       END
134:      -Track'  einlesen; bei XBIOS ist T von 0-79!
135:        XBIOS (Ds%,8, HIGH(Adr0%L), LOW(Adr0%L),0,
             0,0,1,T%-1,1,10)
136:        XBIOS (Ds%,8, HIGH(Adr1%L), LOW(Adr1%L),0,
             0,0,1,T%-1,0,10)
137:        FOR I%=0 TO 256*40:Feld%B(I%)= PEEK(Adr0%L+
             I%): NEXT
138:       RETURN
139:      -Codewdlg'  COMMODORE-Code umwandeln
140:        IF U%<32 THEN IF U%<>13 THEN U%=0
141:        IF U%>64 AND U%<91 THEN U%=U%+32
142:        IF U%>192 AND U%<255 THEN U%=U%-128
143:        IF U%=160 THEN U%=32
144:        IF U%=255 THEN U%=227
145:        IF Zus%L=0 THEN RETURN
146:        'zusätzliche Code-Wandlungen:
147:          IF U%=64 THEN U%=158
148:          IF U%=91 THEN U%=132
149:          IF U%=92 THEN U%=148
150:          IF U%=93 THEN U%=129
151:          IF U%=123 THEN U%=142
152:          IF U%=124 THEN U%=153
153:          IF U%=125 THEN U%=154
154:      RETURN
155:      -Fehler
156:       E$= ERR$ : PRINT
157:       IF E$="TOS error #36" THEN PRINT
             CHR$(27);"B ST-DIRECTORY voll !":
             ELSE PRINT CHR$(27);"B";E$
158:       RESUME Ende
159:      -Ende: END
```

# FILE-INFO

*Uwe Seimet*

JEDER ST-BESITZER KENNT SELBSTVERSTÄNDLICH DIE OPTION "ZEIGE INFO" IM DESKTOP-MENÜ. FÜR EINE AUSGEWÄHLTE DATEI ERSCHEINT BEIM AUFRUF EINE DIALOGBOX, DIE INFORMATIONEN ÜBER DIESE ANZEIGT. DABEI HANDELT ES SICH UM ANGABEN ÜBER DEN DATEINAMEN, ERSTELLUNGSDATUM UND -ZEIT SOWIE ÜBER DAS SCHREIBSCHUTZATTRIBUT. ÄNDERN LASSEN SICH LEIDER NUR DIESES ATTRIBUT UND DER DATEINAME.

Will man jedoch die Erstellungszeit oder andere Attribute ändern, läßt sich nichts machen.

Dabei ist es manchmal sinnvoll, nachträglich das Erstellungsdatum einer Datei ändern zu können, z.B. dann, wenn sich nach einem Kopiervorgang zeigt, daß man wieder einmal vergessen hat, die Uhr im Kontrollfeld zu stellen. Das Ergebnis: Die Datei weist ein unsinniges Erstellungsdatum auf. Bisher blieb in einem solchen Fall nichts anderes übrig, als den Kopiervorgang zu wiederholen (natürlich nach dem Stellen der Uhr), mit einem Disk-Monitor zu drohen oder sich gar ganz geschlagen zu geben.

## Einstellungen

Mit dem Accessory *FILEINFO* ist dieser Ärger nun vorbei. Für die über den Fileselector angewählte Datei erhält man eine ausführlichere Information als das Desktop sie bietet, wobei alle angezeigten Angaben auch geändert werden können. Neben dem Schreibschutzattribut kann eingestellt werden, ob es sich bei der ausgewählten Datei in Zukunft um eine Systemdatei oder um eine versteckte handeln soll. Zur Auswahl der Attribute genügt es, die dafür vorgesehen Buttons in der Dialogbox anzuklicken. Die Einstellung der Buttons beim Betreten der Box entspricht den Attributen, wie sie die Datei im Augenblick des Aufrufs besitzt. Besonders interessant sind die Dateiattribute *System* und *Versteckt*, die vom Desktop aus nicht geändert werden können. Geben Sie einer Datei das Attribut *System*, wird sie nicht mehr auf dem Desktop angezeigt, und auch für den normalen Fileselector ist sie nicht mehr vorhanden. Dennoch kann der Inhalt dieser Datei gelesen werden, denn sie ist nicht wirklich von der Diskette verschwunden. Wozu das gut sein kann? Nun, wenn Sie Disketten besitzen, die hoffnungslos überfüllt sind, geben Sie allen .RSC-Dateien doch einfach den Systemstatus. Danach sieht der Diskketteninhalt schon viel übersichtlicher aus, denn die .RSC-Dateien werden nicht mehr angezeigt. Das Betriebssystem bzw. die Programme, die die .RSC-Dateien lesen sollen, bemerken von der Änderung nichts. Selbstverständlich kann man dieses Verfahren auch auf andere Dateien anwenden, die nur von anderen Programmen nachgeladen werden, also ohnehin nicht über einen Mausklick gestartet werden können. Geben Sie einer Datei das Attribut *Versteckt*, so ist sie für das TOS so gut wie gar nicht mehr zu finden, es sei denn, man geht mit einem Disk-Monitor auf die Suche.




## Kopierschutz

Auf diese Art und Weise kann ein einfacher Kopierschutz für einzelne Dateien realisiert

werden, denn wenn man nicht weiß, daß sich eine Datei auf einer Diskette befindet, kommt man auch nicht auf die Idee, sie kopieren zu wollen. (Selbstverständlich hilft dieses Verfahren nur dann, wenn nicht gleich ein Backup der kompletten Disk gemacht wird.) Daraus resultiert jedoch auch folgende Warnung: Merken Sie sich, welche Dateien Sie versteckt haben, denn nur wenn Sie die Dateinamen kennen, können Sie diese Files wieder sichtbar machen! Hierzu rufen Sie erneut das FILEINFO-Accessory auf, geben den Namen der unsichtbaren Datei über die Tastatur ein (der Fileselector zeigt diesen Namen nicht an!), und voilà: Die Information über die anscheinend nicht vorhandene Datei erscheint auf dem Bildschirm. Ändern Sie nun die File-Attribute entsprechend ab, wird die Datei ab sofort wieder für jedermann sichtbar.

Wichtig ist noch der folgende Hinweis: Wird die Datei, deren Namen oder Attribut Sie soeben geändert haben, in einem Window auf dem Desktop angezeigt, muß die ESC-Taste gedrückt werden, damit die vorgenommenen Änderungen auf das Desktop übertragen werden. Welche Routinen des GEMDOS ermöglichen nun das Holen und Ändern der File-Daten? Für das Holen und Setzen der File-Attribute ist die Funktion *CHANGE MODE* ($31) verantwortlich. Sie ermöglicht das Setzen oder Löschen einzelner Attribute. Erstellungsdatum und -zeit werden mit der Funktion *GSDTOF* ($57) geholt oder gesetzt. Als Parameter wird unter anderem ein Flag übergeben, das bestimmt, ob die Daten ermittelt oder gesetzt werden sollen. Achtung: In der Literatur zum TOS ist dieses Flag oftmals falsch beschrieben. Der Wert muß 0 sein, wenn die Erstellungsdaten geholt werden sollen. Wird eine 1 übergeben, werden diese Daten gesetzt!

Wie sich der Aufruf der oben beschriebenen GEMDOS-Routinen genau gestaltet, kann dem ausführlich kommentierten Quelltext des Programms entnommen werden, das übrigens mit einem zum AS68 kompatiblen Assembler erstellt wurde. Die Objektdaten für die Dialogbox wurden in den Quelltext integriert, so daß keine externe Resource-Datei vorhanden ist. Die Programmlänge nach dem Assemblieren und Linken sollte 2321 Bytes betragen.

PP

```
  1:  *********************************
  2:  * Von Uwe Seimet                *
  3:  * Buchenlochstraße 29           *
  4:  * 6750 Kaiserslautern           *
  5:  * Telefon: 0631/21237           *
  6:  * (c) MAXON Computer GmbH 1990  *
  7:  *********************************
  8:
  9:
 10:  *GEMDOS-Konstanten
 11:  GEMDOS  = 1
 12:  OPEN    = $3d
 13:  CLOSE   = $3e
 14:  CHMOD   = $43
 15:  RENAME  = $56
 16:  GSDTOF  = $57
 17:
 18:  *Wichtige Objektindices
 19:  NAME     =  6
 20:  OK       = 14
 21:
 22:
 23:          text
 24:
 25:          lea stack+400,sp          ;Stackpointer
                                         initialisieren
 26:          move.l #"\*.*",inpath     ;Pfad für Disk
 27:          lea intin,a5              ;Pointer auf
                                         INTIN-Array
 28:          lea intout,a6             ;Pointer auf
                                         INTOUT-Array
 29:          moveq #10,d0              ;appl_init
 30:          move.l #$00010000,d1
 31:          bsr aes
 32:          move (a6),d4              ;apid_nr merken
 33:          move #18,(a5)             ;19 Objekte
 34:  fix:    moveq #114,d0             ;rsrc_obfix
 35:          move.l #$01010100,d1
 36:          bsr aesobj                ;Koordinaten auf
                                         Aufl. umrechnen
 37:          subq #1,(a5)
 38:          bpl fix                   ;nächstes Objekt
 39:          moveq #77,d0              ;graf_handle
 40:          move.l #$00050000,d1
 41:          bsr aes
 42:          move (a6),phys_h      ;phys_handle merken
 43:          moveq #35,d0              ;menu_register
 44:          move.l #$01010100,d1
 45:          move d4,(a5)
 46:          lea name(pc),a0           ;Accessory-Name
 47:          bsr aes         ;in Menüleiste eintragen
 48:          move (a6),nr    ;Nummer des Eintrags merken
 49:  *Warteschleife
 50:  doacc:  moveq #23,d0              ;evnt_mesag
```

```
 51:          move.l #$00010100,d1
 52:          lea ev_buff,a3 ;Puffer für GEM-Messages
 53:          move.l a3,a0
 54:          bsr aes
 55:          cmp #40,(a3)    ;AC_OPEN?
 56:          bne doacc       ;nein-
 57:          move nr,d0
 58:          cmp 8(a3),d0    ;Programm angeklickt?
 59:          bne doacc       ;nein-
 60:          bsr.s act       ;Dialog ausführen
 61:          bra doacc       ;weiter in Warteschleife
 62:
 63:  *Dialog mit dem Benutzer
 64:  act:
 65:          move.l a5,a0    ;Pointer auf INTIN-Array
 66:          moveq #9,d0
 67:  opn:    move #1,(a0)+  ;Parameter für OPEN VIRTUAL
                              WORKSTATION
 68:          dbf d0,opn
 69:          move #2,(a0)
 70:          moveq #100,d0               ;v_opnvwk
 71:          lea contrl+2,a4
 72:          clr.l (a4)+
 73:          move #11,(a4)
 74:          move phys_h,vdi_h
 75:          bsr vdi
 76:          move 6(a4),vdi_h  ;vdi_handle merken
 77:
 78:          moveq #90,d0                ;fsel_input
 79:          move.l #$00020200,d1
 80:          move.l #insel,addrin+4
 81:          lea inpath,a0
 82:          bsr aes          ;Eingabe des Filenamens
 83:          tst 2(a6)        ;Abbruch?
 84:          beq exit         ;ja-
 85:          moveq #2,d0      ;Biene als Mauscursor
 86:          bsr mouse
 87:          lea inpath,a0   ;Pointer auf Pfad
 88:          lea nam,a1   ;Pointer auf alten Filenamen
 89:          lea nnam,a2  ;Pointer auf neuen Filenamen
 90:  tpat:   move.b (a0)+,d0
 91:          move.b d0,(a1)+   ;Pfadnamen kopieren
 92:          move.b d0,(a2)+
 93:          bne tpat
 94:  tpat1:  cmp.l #nam,a1
 95:          bcs exit
 96:          subq.l #1,a2
 97:          cmp.b #"\",-(a1)
 98:          bne tpat1
 99:          addq.l #1,a1
100:          addq.l #1,a2
101:          move.l a2,newnam  ;Pointer auf Beginn des
                                  neuen Namens
102:          lea insel,a0
```

→

```
103:          lea filnam(pc),a2
104:          moveq #10,d0
105:  filspc: move.b #" ",(a2)+ ;Bereich für Namen
                                  löschen
106:          dbf d0,filspc
107:          lea filnam(pc),a2
108:  tsel:   move.b (a0)+,d0   ;Pfad- und Filenamen
                                  zusammensetzen
109:          cmp.b #".",d0
110:          bne.s tsel1
111:          lea filnam+8(pc),a2
112:          bra.s tsel2
113:  tsel1:  move.b d0,(a2)+
114:  tsel2:  move.b d0,(a1)+
115:          bne.s tsel
116:
117:          clr.l -(sp)       ;Attribut ermitteln
118:          pea nam
119:          move #CHMOD,-(sp)
120:          trap #GEMDOS
121:          add #10,sp
122:          tst d0            ;alles klar?
123:          bmi exit          ;dummerweise nicht-
124:          lea flg1+3(pc),a0 ;Pointer auf
125:          lea flg2+3(pc),a1 ;Radio-Buttons
126:          btst #0,d0        ;Read Only?
127:          bne.s ro          ;ja-
128:          exg.l a0,a1
129:  ro:     bclr #0,(a0)      ;Buttons deselektieren
130:          bset #0,(a1)      ;bzw. selektieren
131:          lea flg3+3(pc),a0
132:          lea flg4+3(pc),a1
133:          btst #1,d0        ;versteckt?
134:          bne.s hidden      ;ja-
135:          exg.l a0,a1
136:  hidden: bclr #0,(a0)
137:          bset #0,(a1)
138:          lea flg5+3(pc),a0
139:          lea flg6+3(pc),a1
140:          btst #2,d0        ;System-File?
141:          bne.s system      ;ja-
142:          exg.l a0,a1
143:  system: bclr #0,(a0)
144:          bset #0,(a1)
145:
146:          clr -(sp)
147:          pea nam
148:          move #OPEN,-(sp)  ;Datei muß vor GSDTOF
149:          trap #GEMDOS      ;erst geöffnet werden
150:          addq.l #8,sp
151:          move.l d0,d6      ;Fehler beim Öffnen?
152:          bmi exit          ;ja-
153:          clr -(sp) ;Flag für Datum, Zeit ermitteln
154:          move d6,-(sp)
155:          pea dati        ;Puffer für Datum und Zeit
156:          move #GSDTOF,-(sp)
157:          trap #GEMDOS
158:          add #10,sp
159:          clr.l d1
160:          move dati,d1      ;Zeit holen
161:          lsr #5,d1       ;Sekundenbits rausschieben
162:          move.l d1,d0
163:          and #%111111,d0   ;Minuten isolieren
164:          divu #10,d0
165:          add.b #"0",d0     ;und nach ASCII wandeln
166:          lea time(pc),a0
167:          move.b d0,2(a0)
168:          swap d0
169:          add.b #"0",d0
170:          move.b d0,3(a0)
171:          lsr #6,d1         ;Stunden isolieren
172:          divu #10,d1
173:          add.b #"0",d1
174:          move.b d1,(a0)+
175:          swap d1
176:          add.b #"0",d1
177:          move.b d1,(a0)
178:          clr.l d1
179:          move dati+2,d1          ;Datum holen
180:          move.l d1,d0
181:          and #%11111,d0          ;Tag
182:          divu #10,d0
183:          add.b #"0",d0
184:          lea date(pc),a0
185:          move.b d0,(a0)+
186:          swap d0
187:          add.b #"0",d0
188:          move.b d0,(a0)+
189:          lsr #5,d1
190:          move.l d1,d0
191:          and #%1111,d0               ;Monat
192:          divu #10,d0
193:          add.b #"0",d0
194:          move.b d0,(a0)+
195:          swap d0
196:          add.b #"0",d0
197:          move.b d0,(a0)+
198:          lsr #4,d1                   ;Jahr
199:          add #80,d1
200:          cmp #100,d1 ;falls im Jahr 2000 o. später
201:          bcs.s ovr       ;(man kann ja nie wissen)
202:          sub #100,d1
203:  ovr:    divu #10,d1
204:          add.b #"0",d1
205:          move.b d1,(a0)+
206:          swap d1
207:          add.b #"0",d1
208:          move.b d1,(a0)
209:
210:          moveq #107,d0               ;wind_update
211:          move.l #$01010000,d1
212:          move #1,(a5)
213:          bsr aes
214:          moveq #54,d0                ;form_center
215:          move.l #$00050100,d1
216:          bsr aesobj
217:          movem.l 2(a6),a3/a4         ;form_xy und
                                          form_wh merken
218:          clr d2   ;Bildschirmspeicher reservieren
219:          movem.l a3/a4,2(a5)
220:          bsr form_dial
221:          moveq #42,d0                ;objc_draw
222:          move.l #$06010100,d1
223:          move.l #$00000002,(a5)
224:          move.l 2(a6),4(a5)
225:          move.l 6(a6),8(a5)
226:          bsr aesobj         ;Dialogbox darstellen
227:          clr d0             ;Pfeil als Mauscursor
228:          bsr mouse
229:          moveq #50,d0                ;form_do
230:          move.l #$01010101,d1
231:          move #NAME,(a5)             ;Eingabefeld
232:          bsr aesobj                  ;Dialog starten
233:          move (a6),d7
234:          bclr #15,d7  ;Bit für Doppelklick löschen
235:          cmp #OK,d7   ;OK-Button angewählt?
236:          bne break  ;nein-keine Änderung vornehmen
237:
238:          lea date(pc),a0
239:          clr d0
240:          move.b 4(a0),d0
241:          sub.b #"0",d0
242:          mulu #10,d0
243:          sub.b #"0",d0
244:          add.b 5(a0),d0
245:          sub #80,d0
246:          move d0,d1
247:          asl #4,d1                   ;Jahr
248:          clr d0
249:          move.b 2(a0),d0
250:          sub.b #"0",d0
251:          mulu #10,d0
252:          sub.b #"0",d0
253:          add.b 3(a0),d0
254:          or d0,d1
255:          asl #5,d1                   ;Monat
256:          clr d0
257:          move.b (a0)+,d0
258:          sub.b #"0",d0
259:          mulu #10,d0
260:          sub.b #"0",d0
261:          add.b (a0),d0
262:          or d0,d1                    ;Tag
263:          move d1,dati+2              ;Datum merken
264:          lea time(pc),a0
265:          clr d0
266:          move.b (a0)+,d0
267:          sub.b #"0",d0
268:          mulu #10,d0
269:          sub.b #"0",d0
270:          add.b (a0)+,d0
271:          move d0,d1
```

→

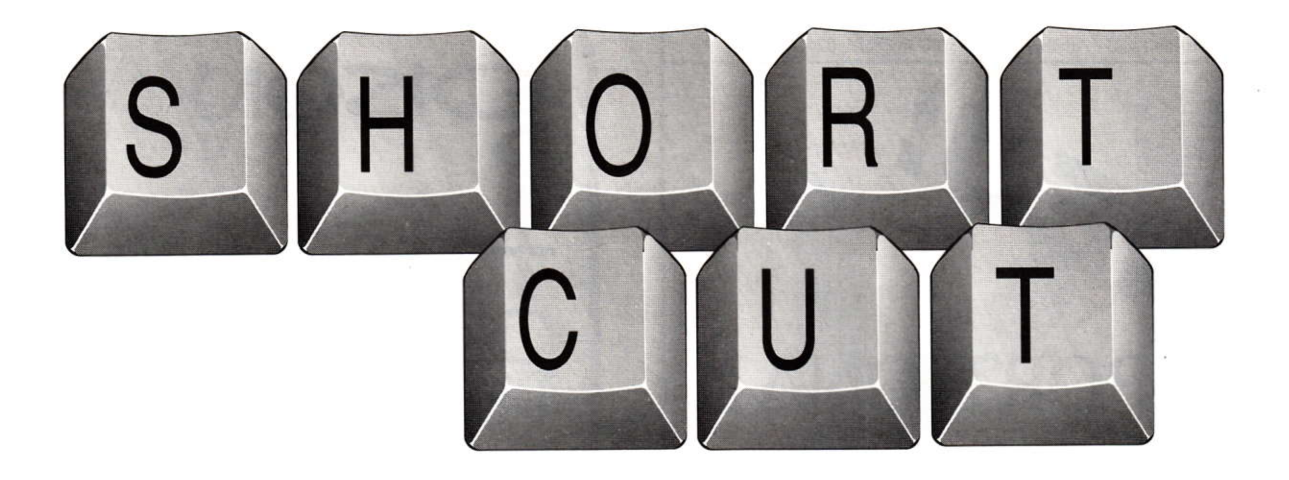
```
272:          asl #6,d1                  ;Stunden
273:          clr d0
274:          move.b (a0)+,d0
275:          sub.b #"0",d0
276:          mulu #10,d0
277:          sub.b #"0",d0
278:          add.b (a0),d0
279:          or d0,d1                   ;Minuten
280:          asl #5,d1
281:          move d1,dati               ;Zeit merken
282:
283:          move #1,-(sp)  ;Datum und Zeit setzen
284:          move d6,-(sp)
285:          pea dati
286:          move #GSDTOF,-(sp)
287:          trap #GEMDOS
288:          add #10,sp
289:
290:          clr d0
291:          btst #0,flg2+3             ;Read Only?
292:          beq.s rol                  ;nein-
293:          bset #0,d0
294:  rol:    btst #0,flg4+3             ;versteckt?
295:          beq.s hidden1              ;nein-
296:          bset #1,d0
297:  hidden1:btst #0,flg6+3             ;System-File?
298:          beq.s system1              ;nein-
299:          bset #2,d0
300:  system1:move d0,-(sp)
301:          move #1,-(sp)
302:          pea nam
303:          move #CHMOD,-(sp)
304:          trap #GEMDOS      ;neue Attribute setzen
305:          add #10,sp
306:
307:          move.l newnam,a1
308:          lea filnam(pc),a0
309:          moveq #7,d0
310:  copnew: move.b (a0)+,d1
311:          cmp.b #" ",d1
312:          beq.s copn2
313:          move.b d1,(a1)+           ;neuen Filenamen
                                         zusammensetzen
314:          beq.s copn1
315:          dbf d0,copnew
316:          bra.s copn3
317:  copn2:  add.l d0,a0
318:  copn3:  move.b #".",(a1)+
319:  copnew1:move.b (a0)+,(a1)+
320:          bne copnew1
321:  copn1:  pea nnam                   ;neuer Name
322:          pea nam                    ;alter Name
323:          clr -(sp)
324:          move #RENAME,-(sp)
325:          trap #GEMDOS               ;Datei umbenennen
326:          add #12,sp
327:
328:  break:  moveq #3,d2 ;Bildschirmspeicher freigeben
329:          movem.l a3/a4,2(a5)
330:          bsr.s fo_dial
331:          moveq #47,d0               ;objc_change
332:          move.l #$08010100,d1
333:          move.l a5,a0
334:          move d7,(a0)+
335:          clr (a0)+
336:          movem.l a3/a4,(a0)
337:          clr.l 12(a5)
338:          bsr.s aesobj ;ausgewählten Button wieder
                            deselektieren
339:          move d7,-(sp)
340:          move #CLOSE,-(sp)
341:          trap #GEMDOS               ;das war's
342:          addq.l #4,sp
343:          moveq #107,d0              ;wind_update
344:          move.l #$01010000,d1
345:          clr (a5)
346:          bsr.s aes
347:
348:  exit:
349:          clr d0              ;Pfeil als Mauscursor
350:          bsr.s mouse
351:          moveq #101,d0              ;v_clsvwk
352:  vdi:    lea contrl,a0
353:          move vdi_h,12(a0)          ;vdi_handle
354:          move d0,(a0)               ;Funktionsnummer
355:          move.l #vdipb,d1
356:          moveq #$73,d0              ;VDI
357:          trap #2                    ;aufrufen
358:          rts
359:
360:  *Mausform auswählen
361:  mouse:
362:          move d0,(a5)       ;Mausform festlegen
363:          moveq #78,d0               ;graf_mouse
364:          move.l #$01010100,d1
365:          bra.s aes
366:
367:  fo_dial:
368:          movem.l a3/a4,10(a5)
369:  form_dial:
370:          moveq #51,d0               ;form_dial
371:          move.l #$09010100,d1
372:          move d2,(a5)
373:
374:  aesobj:
375:          lea _objcdat(pc),a0
376:  aes:
377:          move.l a0,addrin           ;Objekt-Adresse
378:          lea contrl,a0   ;Pointer auf CONTRL-Array
379:          move d0,(a0)               ;Befehlsnummer
380:          movep.l d1,3(a0)           ;Parameter in
                                         Array übertragen
381:          move.l #aespb,d1
382:          move #$c8,d0               ;AES
383:          trap #2                    ;aufrufen
384:          rts
385:
386:
387:  name:   dc.b "  Fileinfo",0        ;Name des
                                         Menüeintrags
388:
389:
390:          even
391:
392:  *----------
393:  *Objektdaten für die Dialogbox
394:  _objcdat:
395:
396:          dc.w $ffff
397:          dc.w $0001,$0012
398:          dc.w $0014
399:          dc.w $0000,$0010
400:          dc.l $00021100
401:          dc.w $0006,$0001
402:          dc.w $0028,$0012
403:
404:          dc.w $0002
405:          dc.w $ffff,$ffff
406:          dc.w $001c
407:          dc.w $0000,$0011
408:          dc.l spec000
409:          dc.w $000c,$0001
410:          dc.w $0010,$0001
411:
412:          dc.w $0003
413:          dc.w $ffff,$ffff
414:          dc.w $0015
415:          dc.w $0000,$0000
416:          dc.l spec001
417:          dc.w $000c,$0003
418:          dc.w $0410,$0001
419:
420:          dc.w $0004
421:          dc.w $ffff,$ffff
422:          dc.w $0015
423:          dc.w $0000,$0000
424:          dc.l spec002
425:          dc.w $000d,$0004
426:          dc.w $020e,$0001
427:
428:          dc.w $0005
429:          dc.w $ffff,$ffff
430:          dc.w $0015
431:          dc.w $0000,$0000
432:          dc.l spec003
433:          dc.w $000d,$0005
434:          dc.w $020e,$0001
435:
436:          dc.w $0006
437:          dc.w $ffff,$ffff
438:          dc.w $0015
439:          dc.w $0000,$0000
```

→

```
440:         dc.l spec004
441:         dc.w $000d,$0006
442:         dc.w $020e,$0001
443:
444:         dc.w $0007
445:         dc.w $ffff,$ffff
446:         dc.w $001d
447:         dc.w $0008,$0000
448:         dc.l spec005
449:         dc.w $000a,$0008
450:         dc.w $0016,$0001
451:
452:         dc.w $0008
453:         dc.w $ffff,$ffff
454:         dc.w $001d
455:         dc.w $0008,$0000
456:         dc.l spec006
457:         dc.w $0002,$0009
458:         dc.w $001a,$0001
459:
460:         dc.w $0009
461:         dc.w $ffff,$ffff
462:         dc.w $001d
463:         dc.w $0008,$0000
464:         dc.l spec007
465:         dc.w $0003,$000a
466:         dc.w $0016,$0001
467:
468:         dc.w $000a
469:         dc.w $ffff,$ffff
470:         dc.w $001a
471:  flg1:  dc.w $0011,$0000
472:         dc.l spec008
473:         dc.w $0002,$000c
474:         dc.w $000b,$0001
475:
476:         dc.w $000b
477:         dc.w $ffff,$ffff
478:         dc.w $001a
479:  flg2:  dc.w $0011,$0000
480:         dc.l spec009
481:         dc.w $000e,$000c
482:         dc.w $000b,$0001
483:
484:         dc.w $000e
485:         dc.w $000c,$000d
486:         dc.w $0019
487:         dc.w $0000,$0000
488:         dc.l $00001100
489:         dc.w $0002,$000e
490:         dc.w $0017,$0001
491:
492:         dc.w $000d
493:         dc.w $ffff,$ffff
494:         dc.w $001a
495:  flg3:  dc.w $0011,$0000
496:         dc.l spec010
497:         dc.w $0000,$0000
498:         dc.w $000b,$0001
499:
500:         dc.w $000b
501:         dc.w $ffff,$ffff
502:         dc.w $001a
503:  flg4:  dc.w $0011,$0000
504:         dc.l spec011
505:         dc.w $000c,$0000
506:         dc.w $000b,$0001
507:
508:         dc.w $000f
509:         dc.w $ffff,$ffff
510:         dc.w $001a
511:         dc.w $0007,$0000
512:         dc.l spec012
513:         dc.w $001e,$000e
514:         dc.w $0008,$0001
515:
516:         dc.w $0012
517:         dc.w $0010,$0011
518:         dc.w $0019
519:         dc.w $0000,$0000
520:         dc.l $00001100
521:         dc.w $0002,$0010
522:         dc.w $0017,$0001
523:
524:         dc.w $0011
525:         dc.w $ffff,$ffff
526:         dc.w $001a
527:  flg5:  dc.w $0011,$0000
528:         dc.l spec013
529:         dc.w $0000,$0000
530:         dc.w $000b,$0001
531:
532:         dc.w $000f
533:         dc.w $ffff,$ffff
534:         dc.w $001a
535:  flg6:  dc.w $0011,$0000
536:         dc.l spec014
537:         dc.w $000c,$0000
538:         dc.w $000b,$0001
539:
540:         dc.w $0000
541:         dc.w $ffff,$ffff
542:         dc.w $001a
543:         dc.w $0025,$0000
544:         dc.l spec015
545:         dc.w $001e,$0010
546:         dc.w $0008,$0001
547:
548:  spec000:dc.b ' FILEINFO V1.00 ',0
549:
550:  spec001:dc.l spec001+$001c,spec001+$0033,spec001+
                    $0034
551:         dc.w $0005
552:         dc.w $0006
553:         dc.w $0000
554:         dc.w $1180
555:         dc.w $0000
556:         dc.w $ffff
557:         dc.w $0017,$0001
558:         dc.b '(C) 1987 by Uwe Seimet',0
559:         dc.b 0
560:         dc.b 0
561:
562:  spec002:dc.l spec002+$001c,spec002+$0030,spec002+
                    $0031
563:         dc.w $0005
564:         dc.w $0006
565:         dc.w $0000
566:         dc.w $1180
567:         dc.w $0000
568:         dc.w $ffff
569:         dc.w $0014,$0001
570:         dc.b 'BuchenlochstraPe 29',0
571:         dc.b 0
572:         dc.b 0
573:
574:  spec003:dc.l spec003+$001c,spec003+$0030,spec003+
                    $0031
575:         dc.w $0005
576:         dc.w $0006
577:         dc.w $0000
578:         dc.w $1180
579:         dc.w $0000
580:         dc.w $ffff
581:         dc.w $0014,$0001
582:         dc.b '6750 Kaiserslautern',0
583:         dc.b 0
584:         dc.b 0
585:
586:  spec004:dc.l spec004+$001c,spec004+$0030,spec004+
                    $0031
587:         dc.w $0005
588:         dc.w $0006
589:         dc.w $0000
590:         dc.w $1180
591:         dc.w $0000
592:         dc.w $ffff
593:         dc.w $0014,$0001
594:         dc.b 'Telefon: 0631/21237',0
595:         dc.b 0
596:         dc.b 0
597:
598:  spec005:dc.l spec005+$001c,spec005+$0028,spec005+
                    $003f
599:         dc.w $0003
600:         dc.w $0006
601:         dc.w $0000
602:         dc.w $1180
603:         dc.w $0000
604:         dc.w $ffff
605:         dc.w $000c,$0017
606:  filnam: dc.b 'FILENAMEEXT',0
```

→

```
607:          dc.b 'Filename: ________.___',0
608:          dc.b 'pppppppppppp',0
609:
610:   spec006:dc.l spec006+$001c,spec006+$0023,spec006+
                $003e
611:          dc.w $0003
612:          dc.w $0006
613:          dc.w $0000
614:          dc.w $1180
615:          dc.w $0000
616:          dc.w $ffff
617:          dc.w $0007,$001b
618:   date:  dc.b '000000',0
619:          dc.b 'Erstellungsdatum: __.__.__',0
620:          dc.b '999999',0
621:
622:   spec007:dc.l spec007+$001c,spec007+$0021,spec007+
                $0038
623:          dc.w $0003
624:          dc.w $0006
625:          dc.w $0000
626:          dc.w $1180
627:          dc.w $0000
628:          dc.w $ffff
629:          dc.w $0005,$0017
630:   time:  dc.b '0000',0
631:          dc.b 'Erstellungszeit: __:__',0
632:          dc.b '9999',0
633:
634:   spec008:dc.b 'Read/Write',0
635:
636:   spec009:dc.b 'Read Only',0
637:
638:   spec010:dc.b 'Sichtbar',0
639:
640:   spec011:dc.b 'Versteckt',0
641:
642:   spec012:dc.b 'OK',0
643:
644:   spec013:dc.b 'Normal',0
645:
646:   spec014:dc.b 'System',0
647:
648:   spec015:dc.b 'ABBRUCH',0
649:   *----------
650:
651:          data
652:
653:   aespb: dc.l contrl,global,intin,intout,addrin,
                addrout
654:
655:   vdipb: dc.l contrl,intin,ptsin,intout,ptsout
656:
657:
658:          bss
659:
660:   *Speicherbereiche für VDI und AES
661:   contrl: ds 11
662:
663:   global: ds 15
664:
665:   intin:  ds 64
666:
667:   ptsin:  ds 64
668:
669:   intout: ds 64
670:
671:   ptsout: ds 64
672:
673:   addrin: ds 64
674:
675:   addrout:ds 64
676:
677:   phys_h: ds 1
678:
679:   vdi_h:  ds 1
680:
681:   nr:     ds 1              ;Nummer des Accessory-
                                 Eintrags
682:
683:   ev_buff:ds 8              ;Puffer für GEM-
                                 Mitteilungen
684:
685:   dati:   ds 2              ;Puffer für Datum und
                                 Zeit
686:
687:   newnam: ds.l 1            ;Pointer auf neuen
                                 Dateinamen
688:
689:          even
690:
691:   inpath: ds.b 40           ;Pfad für Fileselector
692:
693:   insel:  ds.b 13           ;Dateiname für
                                 Fileselector
694:
695:   nam:    ds.b 52           ;kompletter alter Name
                                 der Datei
696:
697:   nnam:   ds.b 52           ;kompletter neuer Name
                                 der Datei
698:
699:          even
700:
701:   stack:  ds.l 100          ;für Stack
```




# Tastengesteuerte Menüleisten

**Kennen Sie das Problem: Sie schreiben ein Programm, das vorschriftsmäßig in GEM eingebunden die Menüleiste zur Genüge benutzt. Ein Programm, das aber schnell bedient werden soll, verlangt vom Programmierer eine Tastensteuerung. Also programmiert man jeden Aufruf, den man als Menüaufruf schon programmiert hat, ein weiteres Mal als Tastaturaufruf (sogenannte Short-Cuts). Abgesehen davon, daß dies eine langweilige und zeitintensive Arbeit ist, verbraucht es auch viel Platz. Und wenn dann ein Menüpunkt eine andere Tastenzuordnung bekommt, müssen wieder das Programm und der Eintrag in der Menüleiste geändert werden. Wenn man aber in der Menüleiste sowieso die Aufruftaste stehen hat, warum soll das Programm diese Zuordnung nicht auch verstehen und automatisch die Tastenzuordnung vornehmen?**

## Das Konzept

Das habe ich mir auch überlegt und ein wenig über das Problem nachgedacht. Prinzipiell müßte man ja nur einmal die Information beim Programmstart aus den Menüleisten herauslesen, jeden Tastendruck mit dieser Informationsmenge vergleichen und bei Übereinstimmung die entsprechende Option auslösen, was sogar über GEM selbst getan werden kann (Überraschung). Bevor ich allerdings auf die Probleme eingehe - der teilweise vorhandene Assemblercode weist darauf hin, daß es doch nicht so einfach war -, beginnen wir doch am Anfang.

## Die Zuordnung

Die Aufgabe ist es, die Informationen, die in der Menüleiste vorhanden sind, auszulesen und in einer Tabelle abzulegen, in der eine Zuordnung zwischen Tastencode und Objektindex des Menüeintrags zu finden ist. Anfangs plante ich zwar, bei einem Tastendruck immer den Menübaum nach einem passenden Eintrag abzusuchen, doch wurde mir schnell bewußt, daß ich zwar dadurch Platz, aber alles andere als Zeit sparen würde. Mit anderen Worten, es galt eine Routine zu entwickeln, die einen kompletten Objektbaum durchwandert und dabei eine weitere Routine aufruft, die überprüft, ob in einem Menüeintrag eine Tastaturdefinition vorhanden ist und daraus die Tabelle bildet. Eine Routine, die einen Objektbaum durchwandert und dabei eine weitere Routine aufruft, war schnell gefunden, da ich eine solche Routine vor ein paar Monaten in der ST-Ecke *Auf Bäume geklettert* der ST-Computer 5/88 veröffentlicht hatte. Diese Routine durchforstet rekursiv den Objektbaum und ruft dabei eine übergebene Routine mit den Parametern Baumadresse und Objektindex auf. In unserem Fall ist es die Routine *set_entry()*, die die Menüeintrag-/Zeichen-Zuordnungstabelle aufbaut.

## Die Definition

Dazu muß natürlich zunächst festgelegt werden, wie man eine Tastaturzuordnung in einer Menüleiste durchführt. Dazu hat sich glücklicherweise ein gewisser Standard durchgesetzt (auch wenn diverse Software-Häuser mal wieder ihre eigenen Wege gehen). Dabei wird die Tastaturzuordnung immer am Ende des Menüeintrags untergebracht. Soll eine Taste in Zusammenhang mit der Control-Taste für einen bestimmten Menüeintrag ausgelöst werden, sagen wir beispielsweise Control-L, schreibt man vor das L ein Dach

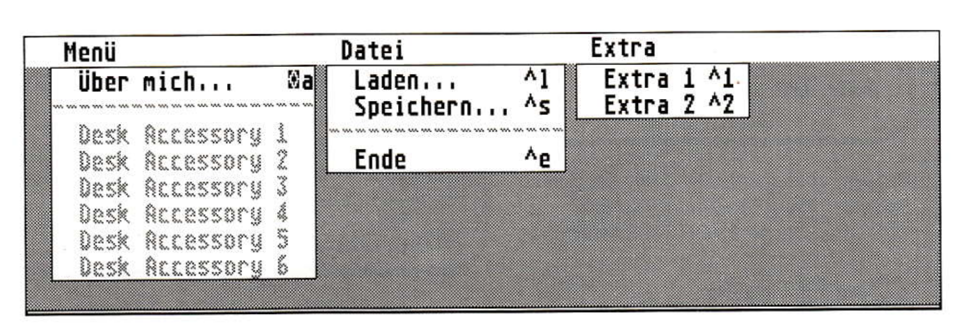



*Bild 1: Menüleiste mit Tastaturbeschriftung*

'^', so daß sich folgende Schreibweise für einen Menüeintrag ergibt: *Lesen einer Datei ^L*. Soll diese Taste in Zusammenhang mit der Alternate-Taste gedrückt werden, so hat sich ▣ als Zeichen standardisiert. Daher werte ich auch diese Zeichen in *set_entry()* aus. Als Beispiel schauen Sie sich bitte Bild 1 an, in dem eine Menüleiste mit definierten Tasten dargestellt ist.

*Set_entry()* ermittelt zunächst die Adresse des zum Objekt gehörenden Strings, wobei in dem hier vorgestellten Fall nur *G_TITLE* und *G_STRING* (ein Menüeintrag) verarbeitet werden. Theoretisch ist es auch möglich, in einer Menüleiste andere Objekte unterzubringen, die durch Tastendruck ausgelöst werden sollen - solche Objektarten sollten eventuell zusätzlich in *set_entry()* berücksichtigt werden. Der Unterschied liegt dann in der Berechnung der Adresse, an der der Text des Objektes zu finden ist. Im folgenden sucht diese Routine vom Ende des Strings ab das erste Vorkommen der CTRL- oder ALTERNATE-Zeichen. Ist ein solches Zeichen gefunden worden, werden dieser Status (CTRL oder ALT) und das Folgezeichen mit dem zugehörigen Objektindex in die Menüeintrag-/Zeichen-Zuordnungstabelle *chr_to_entry[]* eingetragen. Die Rückgabe des Wertes 0 bedeutet nur für *work_tree()*, daß es die Suche nicht abbrechen soll. Das Abbrechen kann in bestimmten Unterroutinen von *work_tree()* von Vorteil sein, ist aber hier nicht erwünscht.

## Das Auffinden

Nachdem nun im Speicher die Tabelle aufgebaut worden ist, brauchen wir noch eine Routine, die überprüft, ob ein Zeichen einem Menüeintrag zugeordnet ist. Diese Aufgabe übernimmt *find_entry()*, das die angelegte Tabelle nach dem übergebenen Zeichen absucht. Hat man das Zeichen in der Tabelle gefunden, wird der Objektindex in einen Puffer eingetragen, der eine größere Menge von anzusprechenden Objekten beinhalten kann. Diese Pufferung hat den Vorteil, daß Tastendrücke, die sich auf die Menüleiste beziehen, auch dann gespeichert werden können, wennn der Rechner erst einmal etwas anderes zu tun hat. Die Routine gibt zurück, ob das Zeichen in der Tabelle gefunden wurde. Dies nutzen wir in der Routine aus, die *find_entry()* aufruft. An dieser Stelle könnte man die Arbeit als schon getan sehen, wenn nicht...

```
 1: /*************************************************/
 2: /*                                               */
 3: /*        Menüleisten-Tastaturansteuerung        */
 4: /*        (Megamax C und Laser C)                */
 5: /*                                               */
 6: /*  Autor: Stefan Höhn     Datum: 15.12.1989     */
 7: /*  Copyright:  MAXON Computer GmbH              */
 8: /*                                               */
 9: /*************************************************/
10:
11: #include <osbind.h>
12: #include <obdefs.h>
13: #include <gemdefs.h>
14: #include "my_menu.h";
15:
16: #define void /**/
17:
18: #define home() Cconout(0x1b);Cconout('H')
19:
20: #define CNTRL_CHAR 0x5e                /* ^ */
21: #define ALT_CHAR   0x07                /* Closebox */
22: #define MAX_ENTRY 99                   /* letzter Eintrag */
23:
24: #define IBUF 0
25: #define IBUFSIZ 4
26: #define IBUFHD 6
27: #define IBUFTL 8
28: #define IBUFLOW 10
29: #define IBUFHI 12
30:
31: /* Long im Tastaturpuffer: 4Bits0/Alt/Cntrl/Rsh/Lsh Scancode(8Bits)
32:                           8Bits0                       ASCII-Wert  */
33: typedef struct
34: {
35:     char  k_status;
36:     char  k_ASCII;
37:     int   k_entry;
38: } KEYCODE_DESCR;
39:
40: typedef struct
41: {
42:     long ibuf;        /* Zeiger auf den Buffer */
43:     int ibufsize;     /* Länge des Buffers */
44:     int ibufhead;     /* nächste Schreibposition */
45:     int ibuftail;     /* nächste Leseposition */
46:     int ibowlow;      /* untere 'Wassermarke' */
47:     int ibufhigh;     /* obere 'Wassermarke' */
48: }IOREC;
49:
50:
51: extern find_entry(), set_entry();
52: extern menu_key();
53: extern intern_buf();
54:
55: extern gl_apid;
56:
57: KEYCODE_DESCR chr_to_entry[MAX_ENTRY+1];  /* Tabelle für Objekte */
58:
59: int act_entry=0;                      /* zeigt auf freien Eintrag */
60:
61: int msg[]={10,0,0,0,1,0,0,0};         /* MN_SELECTED - Ereignis */
62:
63: int main()
64: {
65:     long ssp, ch;
66:     int exit_obj, event, dummy;
67:     int i, cnt=0;
68:     int *poi= (int*)intern_buf;
69:     OBJECT *tree;
70:     IOREC *io_rec;
71:     int msgbuff[8];
72:
73:     io_rec = (IOREC*)Iorec(1);   /* Tastatur */
74:
75:     appl_init();
76:     graf_mouse(ARROW,0L);
77:
78:     if (!rsrc_load("my_menu.rsc"))
79:     {
80:         form_alert(1,"[1][Habe die Resource-Datei|nicht gefunden]
                         [ OK ]");
81:         appl_exit();
82:         return(-1);
83:     }
84:
85:     ssp=Super(0L);
```

## Das Erkennen

Wir haben nun eine Routine, die die Tabelle anlegt, und eine, die die Tabelle auswertet - fehlt nur noch die Routine, die das Zeichen liefert. Wenn da nur nicht das GEM wäre... Die meisten GEM-Applikation, die eine Menüleiste verwalten, verwenden *evnt_multi()* zur Ereignissteuerung, wodurch wir auch automatisch das Tastaturereignis *MU_KEYBD* gewonnen haben. Es hat sich aber sicherlich inzwischen herumgesprochen, daß GEM nur die Applikation mit Zeichen versorgt, die ein Fenster (window) angemeldet und aktiviert hat. Würden Sie also unsere Routinen einsetzen wollen, ohne daß Sie ein Fenster benutzten, so warteten Sie vergeblich auf ein Zeichen durch das Ereignis *MU_KEYBD*. Auch die Idee, mit *Cconis()* auf ein Zeichen zu warten, funktioniert nicht so richtig, da beispielsweise dadurch Zeichen verlustig gehen könnten, die von anderen Routinen, die nebenbei laufen, geklaut würden. Eine Version, bei der ich mit *Cconis()* auf die Zeichen wartete, funktionierte leider in der Hinsicht nicht, daß ich oft Zeichen (mit ALT und CTRL) nicht mitbekam und daher auch nicht darauf reagieren konnte. Als Lösung bleibt eigentlich nur eine Pufferung von Menüleistenereignissen, die allerdings im Interrupt durchgeführt werden muß, in dem ich direkt Zugriff auf die gedrückte Taste habe. Die Aufgabe ist, eine *Interrupt-Service-Routine* zu schreiben, die das aktuell eingegebene Zeichen direkt auswertet, indem sie überprüft, ob dieses Zeichen einer Menüzeilenkodierung entspricht. Ist dies der Fall, wird der entsprechende Objektindex in einen Puffer geschrieben und das Zeichen für die weiteren Applikation nicht mehr weitergegeben.

## Tastaturus interruptus

Das Verfahren der Tastaturabfrage ist ähnlich dem des in der ST-Computer Januar beschriebenen *Knigge* in der Programmierpraxis. Im Unterschied zum Knigge läuft bei uns mal wieder einiges über den Interrupt, um genauer zu sein, über den Tastatur-Interrupt. Bitte schauen Sie sich dazu in Listing 1 die Assembler-Routinen an (die Tastatur-Interrupt-Service-Routine beginnt bei *do_key*). Leider ergibt sich schon am Anfang ein Problem: Biegt man den Tastaturvektor auf eine eigene Routine, bekommt man vom Tastaturprozessor ein Päckchen geschickt, das fein verpackt einen Tastatur-Scancode enthält, der zunächst mit einem ASCII-Wert eines Zeichens relativ wenig am Hut hat! Bevor wir uns aber die Arbeit machen, die Pakete zu entziffern, die Scancodes mühsam in ASCII-Codes umzurechnen und den Ringpuffer zu verwalten (siehe unten), lassen wir dies doch das Betriebssytem tun. Deshalb haben wir in unserer ISR-(Interrupt-Service-Routine) nichts Eiligeres zu tun, als die ISR des Betriebssystems als Unterprogramm aufzurufen (die Adresse haben wir uns gemerkt, bevor wir unsere eigene in den Tastatur-Interrupt-Vektor geschrieben haben). Nachdem das Betriebssystem seine Schuldigkeit getan hat, liegt das Zeichen fertig gestylt im Tastaturpuffer vor und kann seziert werden. Aber Moment: Ist Ihnen die Adresse 0x484 aufgefallen? Sie ist für unsere Anwendung recht wichtig, doch dazu wollen wir uns zunächst anschauen, wie ein Zeichen im Tastaturpuffer aussieht.

## Das Aussehen

Normalerweise würde man annehmen, daß in einem Tastaturpuffer ein Zeichen hintereinander BYTEweise angeordnet ist, aber das wäre nunmal zu einfach gewesen. Immerhin möchten Sondertasten wie F1-F10, HELP, UNDO und so weiter und auch die Steuertasten wie Control, Alternate und Shift dargestellt werden. Das Aussehen eines binärcodierten Zeichens im Druckerpuffer, das vier Bytes lang ist, kann man Tabelle 1 entnehmen.

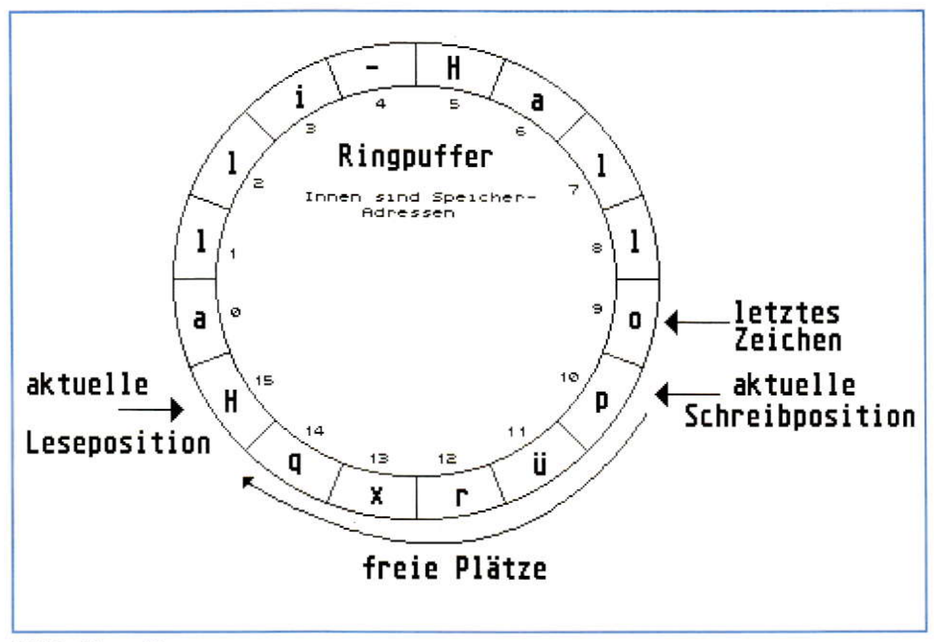



*Bild 2: Ringpuffer*

Dadurch lassen sich alle gewünschten Kombinationen darstellen. Das Problem ist nur, daß das Betriebssystem, faul wie es ist, die Steuertasten leider nur dann einträgt, wenn man es ihm auch wirklich aufträgt. Dies teilt man dem Betriebssystem mit der Betriebssystemvariablen *CONTERM* mit, die an der Adresse 0x484 liegt. Die Einstellungen sehen Sie in Tabelle 2.

Sie sehen also, daß erst dann die Bits 24..31 eingetragen werden, wenn es extra vermerkt wird, was übrigens in der ursprünglichen Einstellung des Betriebssy-

| Bit 31 | Scancode | ASCII Bit0 |
|---|---|---|
| 0000 Alt\| Ctrl\| Rshift\| Lshift | xxxx xxxx 0000 0000 | xxxx xxxx |

*Tabelle 1*

| Conterm (0x484): | |
|---|---|
| | Bit 0 = Tastenklick ein(1)/aus(0) |
| | Bit 1 = Tastenwiederholung ein(1)/aus(0) |
| | Bit 2 = Glocke bei Ctrl-G ein(1)/aus(0) |
| | Bit 3 = Eintragen des Steuertasten-Status' in die Bits 24..31 des Zeichens |

*Tabelle 2*

stems nicht der Fall ist. Nachdem das Betriebssystem uns einen großen Dienst erwiesen hat, sind wir an der Reihe und schauen uns den Tastaturpuffer an, in dem unser Zeichen zu finden ist.

## Das Einsehen

Der Tastaturpuffer arbeitet nach dem Prinzip eines Ringpuffers (siehe Bild 2), das heißt, man definiert einen bestimmten Bereich, in den hineingeschrieben und aus dem herausgelesen wird, wobei man sich Ende und Anfang als verbunden denken muß (deshalb Ringpuffer). Dabei gibt es einen Zeiger, der auf das aktuell zu lesende Zeichen, und einen Zeiger, der auf den nächsten freien Platz für ein neues Zeichen zeigt. Dabei wird beispielsweise beim Betätigen einer Taste hinter dem Zeichen 'o' von "Halli-Hallo" ein weiteres Zeichen angefügt, wobei der Zeiger der nächsten Schreibposition um eins erhöht wird. Soll ein Zeichen gelesen werden, so geschieht das am Anfang der Zeichenkette, denn dort zeigt der Zeiger 'aktuelle Leseposition' hin. Danach wird auch er um eins erhöht. (Kommt der Zeiger an der letzten Speicheradresse des Ringpuffers an, fängt er natürlich wieder von vorne an.) Dabei können zwei Situationen eintreten: Der Puffer ist voll oder leer. Ist er leer, zeigen beide Zeiger auf die gleiche Stelle, während bei einem vollen Puffer der Abstand vom aktuell geschriebenen Zeichen zum aktuellen Lesezeichen genau ein Zeichen beträgt. Die eben erwähnte Vorgehensweise ist die Sicht desjenigen, der ein Zeichen in den Puffer schreiben oder eins aus dem Puffer lesen möchte. In unserem Fall ist aber der Interrupt eine Routine, die das aktuelle Zeichen bearbeiten möchte. Während ein normaler Tastaturleser (nicht Leser dieser Zeitschrift) immer das älteste Zeichen lesen wird (also das 'H' von "Halli..."), möchte die Interrupt-Routine das zuletzt eingegebene und demnach jüngste Zeichen bearbeiten, wodurch sie, nicht wie es sonst üblich ist, am Anfang, sondern am Ende der Zeichenkette lesen wird. Ich erwähne es deshalb, weil man bei der normalen Anwendung, bei der man beispielsweise aus dem Tastaturpuffer einfach vorhandene Zeichen ausliest, anders vorgehen müßte.

Die Information über den Tastaturpuffer liegt im Speicher als Struktur vor, deren Adresse man über die XBIOS-Routine *IOREC()* erfahren kann. Das Ermitteln der Adresse geschieht nicht in der eigentlichen Interrupt-Routine, da man in ihr

```
 86:     init_keyb(1);
 87:     Super(ssp);
 88:
 89:     rsrc_gaddr(R_TREE,MENUE,&tree);
 90:     menu_bar(tree,1);                    /* Setzen der Menüleiste */
 91:
 92:     /* Baum rekurs. durchwandern u.f.jed.Objekt set_entry aufrufen */
 93:     work_tree(tree,0,0,set_entry);
 94:
 95:     msg[1]=gl_apid;                      /* Botschaft an sich selbst */
 96:
 97:     msgbuff[0]=~MN_SELECTED;
 98:     msgbuff[4]=~QUIT;
 99:
100:     while(!(msgbuff[0]==MN_SELECTED && msgbuff[4]==QUIT))
101:     {
102:        event = evnt_multi(MU_MESAG | MU_TIMER,
103:           1,0,1,
104:           0,0,0,0,0,
105:           0,0,0,0,0,
106:           msgbuff,100,0,&dummy,&dummy,&dummy,&dummy,&dummy);
107:
108:        if (Cconis())       /* Zeichen lesen, sonst könnte der Puffer */
109:          ch=Crawcin();     /* bei nicht verwend. Zeichen voll werden */
110:
111:        cnt++;              /* Verzögerung zur Pufferdemonstration */
112:
113:        if(event&MU_TIMER)       /* Timer-Ereignis ? */
114:          if (!(cnt%10))         /* Verzögerung abgelaufen? */
115:             send_menuevnt(); /* Um Menü-Ereignis kümmern */
116:
117:  /* Die folg. Zeilen geben den Puffer als Demostration aus, damit */
118:  /* man sieht wie er abgearbeitet wird. */
119:
120:        home();
121:        printf("\n\n\n\n\n\n\n");
122:        printf("%4x %4x %4x %4x%4x\n",poi[0],poi[1],poi[2],poi[3],
                  poi[4]);
123:        printf("%4x %4x %4x %4x%4x\n",poi[5],poi[6],poi[7],poi[8],
                  poi[9]);
124:        printf("%4x %4x %4x %4x%4x\n",poi[10],poi[11],poi[12],poi[13],
                  poi[14]);
125:        printf("%4x %4x %4x %4x %4x %4x\n",poi[15],poi[16],poi[17],
                  poi[18],poi[19],poi[20]);
126:
127:        printf("Head %3d %2x ->%lx\n",io_rec->ibufhead,
128:              io_rec->ibufhead,*(long*)(io_rec->ibuf+io_rec->ibufhead));
129:        printf("Tail %3d %2x ->%lx\n",io_rec->ibuftail,
130:              io_rec->ibuftail,*(long*)(io_rec->ibuf+io_rec->ibuftail));
131:        if (event&MU_MESAG)
132:           switch(msgbuff[0])
133:           {
134:              case MN_SELECTED:
135:                 default: tree[msgbuff[4]].ob_state^=CHECKED;
136:              menu_tnormal(tree,msgbuff[3],1);
137:              break;
138:           }
139:     }
140:
141:     msgbuff[0]=~MN_SELECTED;
142:     msgbuff[4]=~QUIT;
143:
144:     menu_bar(tree,0);                      /* Löschen der Menüzeile */
145:     rsrc_free();                           /* Freigeben der Resourcen */
146:     appl_exit();
147:
148:     ssp=Super(0L);      /* In  Supervisor schalten */
149:     init_keyb(0);       /* Vektoren zurückbiegen */
150:     Super(ssp);         /* In User-Modus schalten */
151:
152:     return(0);
153:  }
154: /*****************************************************************/
155: /* send_menuevnt: Diese Routine überprüft, ob ein Menüeintrag über */
156: /* Tastatur angewählt worden ist, schickt ein entsprechendes       */
157: /* Ereignis an die eigene Applikation und verschiebt den Inhalt    */
158: /* des internen Puffers.                                  */
159: /*****************************************************************/
160:
161:  send_menuevnt()
162:  {
163:     int *poi= (int*)intern_buf;  /* Zeiger auf den internen Puffer */
164:     int i;
165:
166:     if (poi[0])  /* Ist im Puffer etwas ? */
167:     {
```

keine Betriebssystemroutinen aufrufen darf, sondern im Initialisierungsteil, der die Adresse in *p_iorec* zwischenspeichert. In dieser Struktur finden sich die Adresse des Puffers, die nächste Schreibposition (head), die nächste Leseposition (tail) und zwei weitere Eintragungen, die für den Tastaturpuffer nicht gebraucht werden und Warngrenzen darstellen, wenn der Puffer über oder unter eine gewisse Menge an Zeichen kommt (wird bei der seriellen Schnittstelle verwendet).

## Die Interrupt-Service-Routine

Zunächst überprüft die ISR, ob der interne Puffer, der die Objektindizes enthält, die als Menüpunkte durch die Tastatur ausgelöst wurden, schon voll ist. Ist dies der Fall, wird die ISR sofort verlassen.

Danach wird geprüft, ob unser ISR-Teil aktiv ist, was immer dann der Fall sein wird, wenn *FLAG* gesetzt ist. Fatal ist es nämlich, wenn unsere Routine sehr lange dauern würde und schon das nächste Zeichen ankäme (wie man unten sieht, wird das Interrupt-Level auf 3 gesetzt, so daß ein weiterer Tastatur-Interrupt möglich ist). In diesem Fall würde die ISR durch *FLAG* noch erkennen, daß die ISR noch bearbeitet wird und einfach um den Hauptteil unserer ISR herumspringen. Dies ist außerdem sehr praktisch, wenn Sie bei Änderungen des Hauptteils debuggen wollen. Stellen Sie sich vor, Sie wollten mit einem Debugger eine Tastatur-ISR bearbeiten und setzen mitten hinein einen Breakpoint (Haltpunkt). Das kann einfach nicht funktionieren, da für die Handhabung des Debuggers Zeicheneingaben erforderlich sind, Sie aber mit dem Breakpoint die Zeichenbearbeitung immer wieder anhalten - es ist sogar viel schlimmer, denn das System wird abstürzen, da kein Zeichen zuende bearbeitet wird und die Routine 'total durcheinander' gerät. Wenn Sie aber nur einen für die eigentliche Zeichenbearbeitung nicht so wichtigen Teil, den Sie debuggen wollen, durch ein flag ausklammern, funktioniert es. Es ist praktisch so, daß in dem Moment, wo Sie in Ihren Teil kommen, die Tür hinter sich zumachen, und alle weiteren Interrupts außen vorbei müssen. Wenn Sie fertig sind, machen Sie die Tür wieder auf. Alle Zeichen, die vorbei mußten, können natürlich von Ihrer Routine nicht mehr beabeitet werden.

```
168:        msg[4]=poi[0];          /* Objekt-Nummer in Ereignis eintragen */
169:        appl_write(0,16,msg); /* MN_SELECTED-Ereignis
                                       an GEM schicken */
170:
171:        for(i=1; poi[i] && poi[i]!=0xff; i++) /* Puffer ein Zeichen */
172:           poi[i-1]=poi[i];                   /* verschieben */
173:        poi[i-1]=0;                           /* und letztes Zeichen
                                                     löschen */
174:     }
175: }
176:
177: /*****************************************************************/
178: /* find_entry: Diese Routine übernimmt den Tastencode, zerlegt    */
179: /* ihn in Status und Zeichen und schaut, ob es dieses Zeichen in  */
180: /* der erstellten Menüeintrag/Zeichen-Zuordnungstabelle findet.   */
181: /* Wird das Zeichen gefunden, so schreibt find_entry() die        */
182: /* zugehörige Objektnummer des Menüobjektes i.d. internen Puffer. */
183: /*                                                                */
184: /*****************************************************************/
185: int find_entry(keycode)
186: long keycode;
187: {
188:     register int status,key;
189:     register int ind=0;
190:     register KEYCODE_DESCR *ptr;
191:     register int *poi= (int*)intern_buf;
192:
193:     status=((keycode)>>24)&0x0c;  /* nur Cntrl und Alt interessant */
194:     key   = keycode&0xff;         /* Zeichen */
195:
196:     while (ptr = &chr_to_entry[ind++], ptr->k_entry ) /* gesamte
                                                             Tabelle */
197:        if ( ptr->k_status == status && ptr->k_ASCII == key )
                                                          /* gleich? */
198:        {
199:           /* nächsten freien Eintrag suchen */
200:           for ( ind=0; poi[ind]; ind++ )
201:              if ( poi[ind]==0xff )
202:                 return (0);         /* Liste voll, nicht verwertet */
203:
204:           poi[ind]=ptr->k_entry;   /* Objektindex eintragen */
205:           return (1);              /* Buchstabe verwertet */
206:
207:        }
208:     return(0);                     /* Buchstabe nicht verwertet */
209: }
210:
211: /*****************************************************************/
212: /* set_entry: ... baut die Menüeintrag/Zeichen-Zuordnungstabelle  */
213: /* vom Typ 'Feld von KEYCODE_DESCR' auf, indem es den Menübaum    */
214: /* nach 'entries' absucht und schaut, ob im String Tastenbeschrei-*/
215: /* bungen vorkommen. Diese Beschreibungen werden zusammen mit dem */
216: /* Objektindex in die Tabelle eingetragen. Dadurch kann mit       */
217: /* find_entry() sehr schnell geschaut werden, ob ein Tastencode   */
218: /* sich auf die Menüzeile bezog. Das Aufbauen geschieht in        */
219: /* Zusammenhang mit worktree, das den Menübaum rekursiv           */
220: /* durchläuft und bei jedem Objekt set_entry mit Objektnummer     */
221: /* und Baumadresse aufruft.                                       */
222: /*****************************************************************/
223: int set_entry(baum, ob_ind)
224: OBJECT *baum;
225: int    ob_ind;
226: {
227:     int len;
228:     char *entry_adr, status_char, key_char, status;
229:     int msg[8];
230:
231:     if (act_entry>MAX_ENTRY) /* Schon mehr als MAX_ENTRY+1 Einträge */
232:        return(-1);           /* Abhilfe: MAX_ENTRY vergrößern       */
233:
234:     /* Ist Objekt ein Menüeintrag?
         (Kann auch erweitert werden->ICON...*/
235:
236:     if (baum[ob_ind].ob_type==G_TITLE ||
          baum[ob_ind].ob_type==G_STRING)
237:     {
238:        entry_adr= (char*)baum[ob_ind].ob_spec; /* Adresse auf
                                                      Textstring */
239:        len= strlen(entry_adr);            /* Länge des Textes */
240:
241:        /* Suchen vom Ende an, ob ALT_CHAR o. CNTRL_CHAR gefunden...*/
242:        while(len && *(char*) (entry_adr+len)!=ALT_CHAR && *(char*)
              (entry_adr+len)!=CNTRL_CHAR)
243:           len–;
244:        if (len) /* Suche nicht bis zum Anfang gekommen, d.h. Zeichen
                        gefunden */
```

## *Zeichenverwaltung*

Nun beginnt der eigentliche Zeichen-Handhabungsteil: Die ISR vergleicht *HEAD* und *TAIL* des Ringpuffers und beendet die Bearbeitung, falls der Puffer leer ist. Wie kann so etwas passieren, wenn doch die ISR angesprochen wurde? Über die Tastatur-ISR des Betriebssystems läuft nicht nur die Tastatur, sondern auch die Maus, so daß wir uns dadurch aus der Affaire ziehen. Deshalb wird auch danach noch überprüft, ob sich seit dem letzten Interrupt überhaupt die Schreibposition geändert hat. Ist das nicht der Fall, wurde auch kein neues Zeichen in den Puffer eingetragen (oder es hat sich noch jemand in den Interrupt eingehängt und das Zeichen schon vor uns geklaut). Eine andere Methode zu erkennen, ob es ein Tastaturereignis war, daß den Interrupt ausgelöst hat, ist, das Päckchen zu untersuchen, das vom Tastaturprozessor geschickt wurde. Allerdings wollten wir ja auf dieser Ebene nicht arbeiten, sondern solche 'niederen' Arbeiten dem Betriebssystem überlassen. Sollte Sie die Päckchenverarbeitung interessieren, verweise ich Sie auf die ST-Ecke *Quick-Mouse, eine Mausroutine für den ST* in der ST-Computer 2/88. Jetzt holen wir unser aktuelles Zeichen aus dem Puffer, das leider noch ein wenig bearbeitet werden muß. Falls nämlich Shift oder CAPS-LOCK gedrückt waren, während das Zeichen eingelesen wurde, muß dies noch berücksichtigt werden. Außerdem hat man das Problem, daß bei ALT/CTRL-Tasten oft zum Scancode kein ASCII-Code eingetragen wird, ohne den wir aber nicht arbeiten können. Dazu existieren drei Konvertierungstabellen, deren Adressen man durch *Keytbl()* mitgeteilt bekommt (oder auch auf eigene Adressen setzen kann). Die Umkodierung ist relativ einfach, indem man mit dem Scancode in die Tabellen hineinindext und damit das entsprechende Zeichen erhält.

Jetzt folgt Trick 17: Wir rufen mit diesem Zeichen unsere C-Routine(!) *find_entry()* auf, die, wie oben beschrieben (aus unserer Interrupt-Service-Routine heraus), prüft, ob dieses Zeichen einem Menüeintrag entspricht, und entsprechend den Objektindex in den internen Puffer einträgt. Da *find_entry()* zurückgibt, ob das Zeichen verwendet wurde, kann auch die ISR entscheiden, ob das Zeichen aus dem Puffer entfernt werden soll. Dies geschieht dadurch, daß der tail des Ringpuffers einfach wieder um ein Zeichen zurückgesetzt wird. Damit wäre dann auch der Job der ISR getan.

```
245:
246:          {
247:             status_char= *(char*) (entry_adr+len); /* CNTRL- oder ALT-
                                                          Buchstabe */
248:             key_char    = *(char*) (entry_adr+len+1); /* Zeichen */
249:
250:             status = (status_char==ALT_CHAR)<<3 |
                          (status_char==CNTRL_CHAR)<<2;
251:
252:             /* Tabelleneintrag erstellen */
253:             chr_to_entry[act_entry].k_status=status; /* ALT o. CNTRL */
254:             chr_to_entry[act_entry].k_ASCII=key_char; /* Zeichen */
255:             chr_to_entry[act_entry].k_entry=ob_ind;   /* Objekt-Index */
256:             chr_to_entry[++act_entry].k_entry=0;    /* Endemarkierung */
257:          }
258:       }
259:       return(0); /* weiter im Baum suchen */
260:  }
261:  /******************************************************************/
262:  /* work_tree: Diese Routine wandert den Objektbaum vollständig ab. */
263:  /*            Ihr wird eine Routine übergeben, die automatisch bei */
264:  /*            jedem Objekt aufgerufen wird. So könnte man bei-     */
265:  /*            spielsweise einen ganzen (Unter-)Baum selektieren    */
266:  /*            oder je nach Eigenschaft disablen...                 */
267:  /*            Die aufgerufene Routine bekommt Adresse des Baumes   */
268:  /*            und die Objektnummer übergeben.                      */
269:  /******************************************************************/
270:  void work_tree(baum, ob_an, ob_en, aufruf)
271:  OBJECT *baum;
272:  int ob_an, ob_en;
273:  int (*aufruf)();
274:  {
275:     int ob_ind; /* Objekt-Index */
276:
277:
278:     ob_ind=ob_an; /* Von erstem Objekt an... */
279:
280:     do
281:     {
282:        if ((*aufruf)(baum,ob_ind)==-1) /* angegeb. Routine aufrufen */
283:           break;                      /* bei Rückgabe=-1, abbrechen */
284:
285:        if (baum[ob_ind].ob_head!=-1)     /* Baumverzeigung erkannt */
286:        {                                 /* Unterbaum abarbeiten */
287:           work_tree(baum, baum[ob_ind].ob_head,
                         baum[ob_ind].ob_tail, aufruf);
288:        }
289:        if (ob_ind==ob_en)             /* Am letzten Objekt angekommen */
290:           break;
291:        ob_ind=baum[ob_ind].ob_next;      /* nächstes Objekt */
292:     }while (baum[ob_ind].ob_next!=1);   /* am Ende */
293:  }
294:
295:  #define illegal dc.w 0x4afc /* prakt. z.Debuggen, springt bspw. */
296:                              /* in den Tempelmon oder SL-Debugger */
297:
298:  /****************************************************************/
299:  /* init_keyb: Das Argument gibt an, ob die Vektoren installiert  */
300:  /* oder wieder restauriert werden sollen. Hierin ist der         */
301:  /* komplette Tastaturtreiber eingebunden. Er überprüft direkt    */
302:  /* während des Interrupts, ob ein Zeichen in der Menüeintrag-/   */
303:  /* Zeichen-Zuordnungstabelle zu finden ist, momentan eingegeben  */
304:  /* wird. Ist dies der Fall, schreibt er den entsprechenden       */
305:  /* Objektindex in den internen Objektindexpuffer intern_buf.     */
306:  /* Auf diesen Puffer hat man dann extern Zugriff.                */
307:  /*                                                               */
308:  /* Achtung: Auf keinen Fall Programm verlassen, ohne vorher      */
309:  /* installierte Vektoren wieder restauriert zu haben!            */
310:  /* Das Retten von A4 ist nur für die alte Megamax-C-Version      */
311:  /* nötig, da hier die globalen Variablen über A4 adressiert      */
312:  /* werden. Laser-C adressiert absolut, so daß A4 keine           */
313:  /* Rolle spielt.                                                 */
314:  /****************************************************************/
315:
316:  void init_keyb(arg)  /* 1: installieren, 0: restaurieren */
317:  char arg;
318:  {
319:     static int flag();
320:     static save_keyb(), s_conterm(), do_key(), p_iorec(),
321:            save_a4(), last_tail(),k_tabvec();
322:     int ch;
323:
324:     while(Cconis())       /* Solange etwas im Tastaturpuffer ist */
325:        ch=Crawcin();      /* Zeichen auslesen */
326:
327:     asm{
```

## Die Anwendung

Alles schön und gut: Jetzt haben wir (interruptgesteuert und gepuffert) einen Speicherbereich, in dem aufgereiht die Objektindizes auf ihre Abarbeitung warten. Wie aber bringen wir sie zur Auswertung? Ganz einfach: Wir programmieren *evnt_multi()* mit einem Timer-Ereignis *MU_TIMER* und einer Zeit von etwa 100ms. Jedesmal, wenn dieses Ereignis ausgelöst wurde, schaut *send_menuevnt()* nach, ob im internen Puffer ein Objektindex steht, der auf Bearbeitung wartet. Ist dies der Fall, tritt Trick 18 in Kraft: Die Applikation schickt sich selbst ein *MU_MESAG*-Ereignis mit der *MN_SELECTED*-Botschaft des betreffenden Objektes über den Befehl *appl_write*. Dies funktioniert ganz einfach dadurch, daß man als Absender die *AP*(plication)*_ID*entification des eigenen Programmes angibt und den Ereignispuffer mit den entsprechenden Werten auffüllt (siehe ST-Ecke *Talkshow...* ST-Computer 1/88). Ist die durch *MU_TIMER* ausgelöste *multi_evnt()*-Schleife bearbeitet, endet sie im Endeffekt wieder in *evnt_multi()*, mit dem Erfolg, daß dieser nichts Besseres zu tun hat, als sofort zu bemerken, daß ein *MU_MESAG* mit *MN_SELECTED* angemeldet worden ist, und damit eine weitere *evnt_multi()*-Schleife initiiert....

## Weiterführendes

Einige Leser unter Ihnen werden schon bemerkt haben, daß man an diesen Routinen noch einiges an Erweiterungen unterbringen kann, wobei ich auf eine Sache hinweisen möchte. Mit dieser Form von *find_entry()* ist es nur möglich, mit CTRL- oder ALT-Kombinationen mit normalen Tasten Menüs auszulösen. Deshalb wäre es interessant, diese Routine dahingehend zu erweitern, daß sie auch die Sondertasten (undo, clr/home, F1 - F10 etc.) oder Tasten ohne ALT/CTRL-Deklarierung interpretieren kann. Sollte Ihnen das Konzept gefallen, können Sie sich jedenfalls sehr viel Arbeit sparen. Stellen Sie sich vor, Sie müßten die gesamte Menüleisten-Tasten-Zuordnung ändern: Kein Problem! Sie gehen einfach in das Resource-Construction-Set, ändern die Texteinträge, und schon läuft das Programm mit anderen Tastenkombinationen...

*SH*

```
328:          movem.l D0-D2/A0-A2, -(A7) /* Register retten */
329:
330:          lea save_a4,   A1    /* Platz zum Retten */
331:          move.l A4 , (A1)     /* Rette Register für globale Variablen */
332:
333:          move.w SR,-(A7)      /* Statusregister retten */
334:          ori.w  #0x700,SR     /* alle interrupts sperren */
335:
336:          tst.b  9(A6)         /* arg = 0 */
337:          bne    begin_keyb    /* Nein -> VBL-Routine initialisieren */
338:
339:  end_keyb:
340:
341:          move.w #34,-(A7)            /* Xbios(34) == KBDVBASE */
342:          trap #14                    /* Xbios-Aufruf */
343:          addq.l #2, A7               /* Stack korrigieren */
344:          move.l D0,A0                /* Routine table vector */
345:          move.l save_keyb,32(A0)     /* alte Adresse wiederherstellen */
346:
347:          bclr  #3,0x484              /* conterm shift status rücksetzen */
348:          move.b s_conterm,D0         /* Ist es gleich Null? */
349:          beq   end_init              /* JA */
350:          bset  #3,0x484              /* conterm-shift-Status setzen */
351:
352:          bra end_init                /* Ende der Installierung */
353:
354:  begin_keyb:
355:
356:          move.w #34,-(A7)            /* Xbios(34) == KBDVBASE */
357:          trap #14                    /* Xbios-Aufruf */
358:          addq.l #2, A7               /* Stack korrigieren */
359:          move.l D0,A0                /* Routine table vector */
360:          lea    save_keyb, A1        /* Adresse zum Speichern des alten
                                             Zeigers */
361:          move.l 32(A0), (A1)         /* alte Adresse abspeichern */
362:
363:          lea   do_key, A1        /* neuen keyboard handler eintragen */
364:          move.l A1, 32(A0)
365:
366:          move.w  #1,-(A7)           /* Gerät: Tastatur */
367:          move.w  #14,-(A7)          /* Xbios(14) == IOREC */
368:          trap    #14                /* Xbios-Aufruf */
369:          addq.l  #4, A7             /* Stack korrigieren */
370:          lea     p_iorec, A1        /* Adresse der Iorec-Struktur */
371:          move.l  D0, (A1)           /* zwischenspeichern */
372:          lea     last_tail, A0      /* Adresse:aktuelle Schreibposition */
373:          move.w  IBUFTL(A1), (A0)   /* Position merken  */
374:
375:          pea    -1                   /* Tabellenaddressen nur erfragen */
376:          pea    -1
377:          pea    -1
378:          move.w #16,-(A7)            /* Xbios(16) == KEYTBL */
379:          trap #14                    /* Xbios-Aufruf */
380:          lea    14(A7), A7           /* Stack korrigieren */
381:          lea   k_tabvec, A1          /* Adresse der Keytable-Vektoren */
382:          move.l D0, (A1)             /* zwischenspeichern */
383:
384:          lea    s_conterm,A1
385:          btst  #3,0x484              /* conterm shift status testen */
386:          sne    (A1)
387:
388:   end_init:
389:          move.w (A7)+,SR             /* interrupts wieder freigeben */
390:          bra    ende
391:
392:
393:
394:   save_a4:    dc.l  0    /* Platz für A4, globale Variablen-Referenz */
395:   p_iorec:    dc.l  0    /* Zeiger auf Puffer-Struktur */
396:   k_tabvec:   dc.l  0    /* Adresse der Keyboard-Vektor-Tabelle */
397:   s_conterm:  dc.b  0    /* Shift-Status-Merker */
398:
399:   /* im Weiteren folgt die XBRA-Struktur */
400:   dc.l 0x58425241                  /* 'XBRA' */
401:   dc.l 0x4D656E81                  /* 'Menü' */
402:   save_keyb: dc.l  0               /* Adresse der alten Routine */
403:
404:
405:
406:   do_key:
407:          movem.l  D0-D2/A0-A2, -(A7)
408:
409:          bset      #3,0x484     /* conterm Shift-Status immer setzen! */
410:          move.l    save_keyb,A1
411:          jsr       (A1)         /* Original-Keyboard Handler ZUERSCHT! */
412:
```

```
413:        lea     intern_buf, A1 /* Zeiger auf
                                   internen Puffer */
414:        move.w  38(A1), D0     /* letzter Wert */
415:        bne     int_quit       /* Puffer voll!
                                   -> Schluß */
416:
417:        move.w flag, D0  /* Flag sagt, ob
                             unsere Routine aktiv */
418:        cmp.w   #0xff, D0 /* aktiv? */
419:        beq     int_quit  /* Dann umgehen... */
420:        lea     flag, A1  /* War nicht aktiv,
                              jetzt aber */
421:        move.w #0xff,(A1)  /* Praktisch zum
                              Debuggen! */
422:
423:        move.w SR, -(A7)    /* Statusreg. retten */
424:        ori.w #0x0700, SR   /* Interrupt-level 7 */
425:
426:        bclr    #6,0xfffffa11  /* lösche int
                                   service bit */
427:        andi.w  #0xF3FF,SR     /* Int-Maske zurück
428:                                  auf 3 */
429:        move.l p_iorec, A0     /* Adresse der
                                  IOREC-Struktur */
430:        move.w IBUFHD(A0), D1  /* ibufhead: nächste
                                  Schreibposition */
431:        move.w IBUFTL(A0), D2  /* ibuftail: nächste
                                  Leseposition */
432:        cmp.w   D1, D2  /* head = tail? dann kein
                                   Zeichen da */
433:        beq     not_end3    /* nix wie raus hier ! */
434:
435:        move.w last_tail, D0   /* alte Schreib-
                                  position */
436:        cmp.w   D0, D2  /* mit neuer vergleichen */
437:        beq     not_end3    /* Interrupt kam nicht
                                von Tastatur */
438:
439:    not_end1:              /* D1 enthält neuen Head */
440:        move.l (A0), A1     /* Beginn des Puffers */
441:        move.l 0(A1,D2.w), D0  /* letztes (!)
                         Zeichen aus Puffer holen */
442:        move.w D2, D1   /* Tail zwischenspeichern */
443:
444:        clr.w D0           /* unteres Wort löschen */
445:        swap  D0           /* oberes Wort nach unten */
446:        move.l k_tabvec, A2 /* Adresse auf
                         Konvertierungstabellen */
447:
448:        move.l 4(A2), A1    /* Tabelle für Shift */
449:        btst #8, D0         /* linke Shift-Taste */
450:        bne  its_shift      /* ja */
451:        btst #9, D0         /* rechte Shift-Taste */
452:        bne  its_shift      /* ja */
453:        move.l (A2), A1     /* sonst, keine Shift-
                               Tasten-Tabelle */
454:
455:    its_shift:
456:        moveq  #127, D2     /* Maske nach D2 */
457:        and.b  D0, D2      /* Bit 7 ausblenden */
458:        move.b 0(A1,D2.w), D2   /* Zeichen
                                 kovertieren */
459:        swap   D0 /* Zeichencode wieder umdrehen */
460:        or.w   D2, D0  /* ASCII unten einblenden */
461:
462:        movem.l  D1-D7/A0-A6, -(A7) /* Register
                                      retten */
463:
464:        move.l  save_a4, A4 /* A4 für Global-
                                Variablen */
465:        move.l  D0,-(A7)     /* Zeichen auf Stack,
                           find_entry-Parameter */
466:
467:        jsr find_entry  /* laßt es uns auswerten */
468:        addq.l #4, A7   /* Stack korrigieren */
469:
470:        movem.l (A7)+,D1-D7/A0-A6 /* Register
                                    holen */
471:
472:        tst.w  D0   /* return: Zeichen verwertet? */
473:        beq.s  not_end3  /* Nein, Puffer in Ruhe
                               lassen */
474:                        /* Ja, Zeichen löschen */
475:    not_end2:
476:        move.w D1, IBUFTL(A0) /* neuen Headindex
                              sichern; damit */
477:              /* gelesenes Zeichen entfernen */
478:    not_end3:
479:        lea last_tail, A1  /* Speicheradresse für
                              Tail */
480:        move.w IBUFTL(A0), (A1)  /* aktuellen Tail
                                  merken */
481:
482:        lea flag, A1       /* Interrupt-Service  */
483:        move.w #0, (A1)   /* nicht mehr aktiv */
484:    int_ende:
485:        bset     #6,0xfffffa11  /* setze int
                                  service bit */
486:        move.w (A7)+, SR  /* alten Status retten */
487:    int_quit:
488:        movem.l (A7)+,D0-D2/A0-A2 /* Register
                                   restaurieren */
489:        rts                      /* Ciao */
490:
491:
492:    last_tail:  dc.w 0              /* alter Tail */
493:    flag:       dc.w 0              /* ISR-Flag */
494:
495:    intern_buf: dc.w 0,0,0,0,0,0,0,0,0,0
496:                dc.w 0,0,0,0,0,0,0,0,0,0,0xff
                    /* 20+1 Menüeinträge */
497:
498:    ende:
499:      movem.l (A7)+,D0-D2/A0-A2
                  /* Ende der (Re)Installierung */
500:
501:          }
502:    }
```




# GRÜSSE AUS WALHALLA TEIL 3

**Heute stellen wir das Wodan-Modul vor, das die Verwaltung der Dialogboxen übernimmt. *HandleWodan* ermöglicht die Initialisierung des Accessories, das Abspeichern und Laden von Parameterdateien sowie die Verwaltung der beiden Dialogboxen oder besser die Durchführung der Dialoge.**

Die Initialisierung des Accessories geschieht in der Routine *InitAcc*. Zuerst wird das GEM mit *InitGem* initialisiert. Es würde jedoch auch eine Anmeldung beim AES via *appl_init* (AES 10) genügen. Für den Fall, daß keine GEM-Initialisierung erfolgen kann - z. B., wenn *appl_init* den Wert -1 zurückgibt - wird die Ausführung der Initialisierung terminiert. Ansonsten erfolgt eine Eintragung in das Menü über *RegisterAcc*, gefolgt von einer Terminierung für den Fall, daß keine Eintragung erfolgen kann. Die letzte Aktion, die einen Abbruch der Initialisierung hervorrufen kann, ist das Laden der Resource-Datei. Wurde sie nicht gefunden, ist eine weitere Ausführung des Programms nicht sinnvoll. Schließlich werden noch die Adressen der einzelnen Objektbäume ermittelt und diverse Werte gesetzt.

Die Prozedur zum Lesen der Parameter *ReadParameter* beinhaltet auch die Standardparameter, die gesetzt werden, wenn beim Laden der Datei ein Fehler aufgetreten ist. *ReadParameter* wird von *LoadParameter* aufgerufen, wobei *LoadParameter* lediglich den Dateinamen bestimmt. Auch die erste Parameterinitialisierung *InitParameter* ruft *ReadParameter* auf. Da hier der Parametername bereits feststeht - WODAN.INF -, braucht natürlich kein Name gewählt zu werden. *UpdateWindow* (oder in C: *wind_update* mit END_UPDATE/BEG_UPDATE) muß vor jedem Dialog aufgerufen werden, damit nicht irgendein Menü oder Window die Dialogbox zerstören kann.

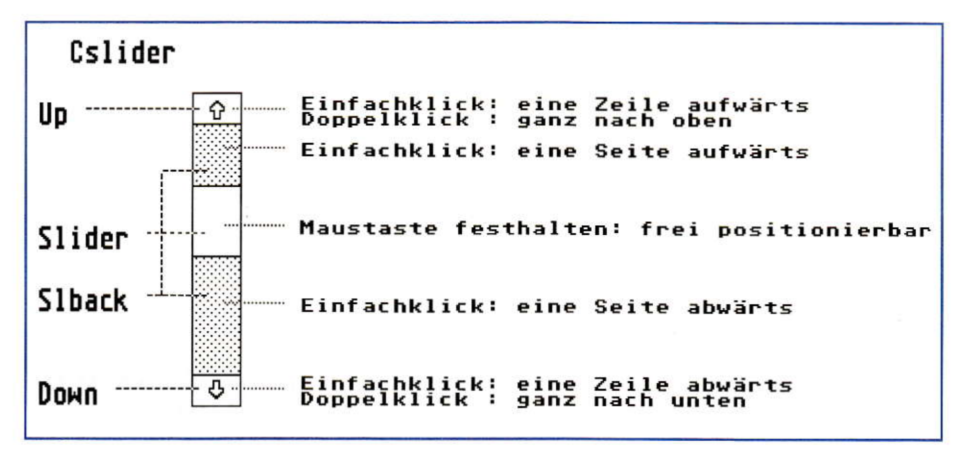



*Bild 1: Die Funktionen des Wodan-Sliders*

*HandleStrings* ist die Prozedur, die beide Dialoge von Wodan durchführt. Der Hilfe-Dialog erscheint gerade im Vergleich mit dem Rest der Prozedur als sehr knapp. Er ist komplett in der Unterprozedur *DoHelp* untergebracht. Der Rest befaßt sich dann mit dem Auswahldialog. *Draw-Slider* gibt den Slider erneut aus, wobei die Position korrekt gesetzt wird. Es ist übrigens nicht nötig, hier Float-Routinen zu verwenden, die Berechnungen erfolgen ausschließlich über Ganzzahltypen. Die Unterprozedur *OutStrings* setzt alle Strings in der Dialogbox und zeichnet diese bei Bedarf auch neu.

Nun aber zur eigentlichen Verwaltung des Dialogs. Nach vorbereitenden Einstellungen, die die Rettung der Einstellungen beinhalten, folgt eine Schleife, die nur dann verlassen wird, wenn "OK" oder "Abbruch" gewählt wurde. Die Einstellungen müssen gerettet werden, damit die Werte später zurückgesetzt werden können, sofern "Abbruch" gewählt wurde. "Abbruch" beinhaltet immer die Funktion, daß alte Einstellungen zurückgesetzt werden. Innerhalb der Schleife müssen weitere Vorbereitungen getroffen werden. Dies sind Einstellungen, die auch durch das Laden einer Parameterdatei Veränderungen hervorrufen können: Status von "an/aus" setzen, Slider positionieren und Zeichenketten ausgeben.

Nachdem diese Vorbereitungen getroffen wurden, erfolgt die Ausgabe der Dialog-




# GRÜSSE AUS WALHALLA
## TEIL 3

**Heute stellen wir das Wodan-Modul vor, das die Verwaltung der Dialogboxen übernimmt. *HandleWodan* ermöglicht die Initialisierung des Accessories, das Abspeichern und Laden von Parameterdateien sowie die Verwaltung der beiden Dialogboxen oder besser die Durchführung der Dialoge.**

Die Initialisierung des Accessories geschieht in der Routine *InitAcc*. Zuerst wird das GEM mit *InitGem* initialisiert. Es würde jedoch auch eine Anmeldung beim AES via *appl_init* (AES 10) genügen. Für den Fall, daß keine GEM-Initialisierung erfolgen kann - z. B., wenn *appl_init* den Wert -1 zurückgibt - wird die Ausführung der Initialisierung terminiert. Ansonsten erfolgt eine Eintragung in das Menü über *RegisterAcc*, gefolgt von einer Terminierung für den Fall, daß keine Eintragung erfolgen kann. Die letzte Aktion, die einen Abbruch der Initialisierung hervorrufen kann, ist das Laden der Resource-Datei. Wurde sie nicht gefunden, ist eine weitere Ausführung des Programms nicht sinnvoll. Schließlich werden noch die Adressen der einzelnen Objektbäume ermittelt und diverse Werte gesetzt.

Die Prozedur zum Lesen der Parameter *ReadParameter* beinhaltet auch die Standardparameter, die gesetzt werden, wenn beim Laden der Datei ein Fehler aufgetreten ist. *ReadParameter* wird von *LoadParameter* aufgerufen, wobei *LoadParameter* lediglich den Dateinamen bestimmt. Auch die erste Parameterinitialisierung *InitParameter* ruft *ReadParameter* auf. Da hier der Parametername bereits feststeht - WODAN.INF -, braucht natürlich kein Name gewählt zu werden. *UpdateWindow* (oder in C: *wind_update* mit END_UPDATE/BEG_UPDATE) muß vor jedem Dialog aufgerufen werden, damit nicht irgendein Menü oder Window die Dialogbox zerstören kann.

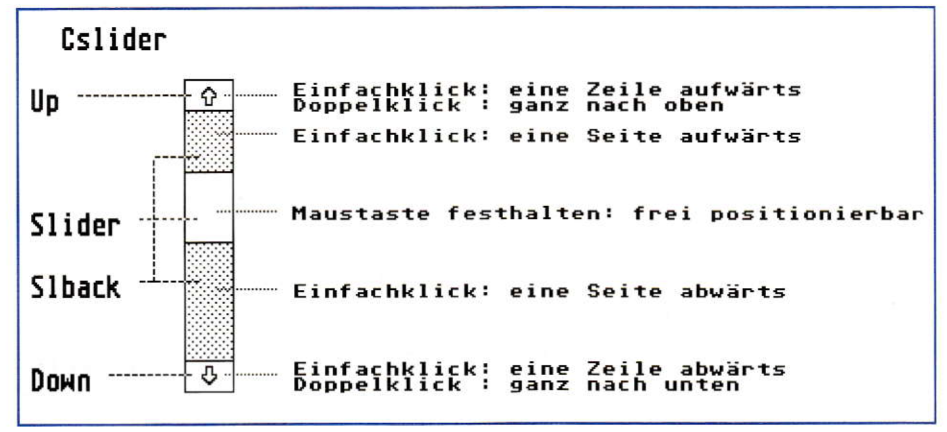



*Bild 1: Die Funktionen des Wodan-Sliders*

*HandleStrings* ist die Prozedur, die beide Dialoge von Wodan durchführt. Der Hilfe-Dialog erscheint gerade im Vergleich mit dem Rest der Prozedur als sehr knapp. Er ist komplett in der Unterprozedur *DoHelp* untergebracht. Der Rest befaßt sich dann mit dem Auswahldialog. *Draw-Slider* gibt den Slider erneut aus, wobei die Position korrekt gesetzt wird. Es ist übrigens nicht nötig, hier Float-Routinen zu verwenden, die Berechnungen erfolgen ausschließlich über Ganzzahltypen. Die Unterprozedur *OutStrings* setzt alle Strings in der Dialogbox und zeichnet diese bei Bedarf auch neu.

Nun aber zur eigentlichen Verwaltung des Dialogs. Nach vorbereitenden Einstellungen, die die Rettung der Einstellungen beinhalten, folgt eine Schleife, die nur dann verlassen wird, wenn "OK" oder "Abbruch" gewählt wurde. Die Einstellungen müssen gerettet werden, damit die Werte später zurückgesetzt werden können, sofern "Abbruch" gewählt wurde. "Abbruch" beinhaltet immer die Funktion, daß alte Einstellungen zurückgesetzt werden. Innerhalb der Schleife müssen weitere Vorbereitungen getroffen werden. Dies sind Einstellungen, die auch durch das Laden einer Parameterdatei Veränderungen hervorrufen können: Status von "an/aus" setzen, Slider positionieren und Zeichenketten ausgeben.

Nachdem diese Vorbereitungen getroffen wurden, erfolgt die Ausgabe der Dialog-

box und wieder eine Schleife. Diese Schleife wird nur dann beendet, wenn ein Objekt angeklickt wurde, welches eine Beendigung des Dialogs zur Folge hat, d. h. solange die Dialogbox zu sehen ist, verbleibt das Programm in dieser Schleife. Nach dem Aufruf von *FormDo*, der Dialogverwaltung des AES, folgen die diversen eigenen Verwaltungsvorgänge, zu denen zunächst die Ermittlung des angeklickten Exit-Objekts und die Auswertung des Doppelklicks gehören. Anschließend werden die möglicherweise veränderten Zeichenketten abgespeichert.

Was nun noch innerhalb der inneren Schleife folgt, ist die Bedienung des Sliders. Für den Fall, daß der Slider selbst angeklickt wurde, muß dieser innerhalb seines Parent-Objekts bewegt werden. Das AES stellt hierzu die Funktion *SlideBox* zur Verfügung, die einen Wert zwischen 0 und 1000 je nach Stellung des Sliders zurückliefert. Es ist aber unbedingt darauf zu achten, daß das Accessory die Mauskontrolle übernimmt, da sonst das Programm hängenbleibt, sobald ein Wodan-Benutzer den Slider bewegen will.

Bei einem Klick auf den Slider-Hintergrund muß vor der Versetzung des Sliders ermittelt werden, ob ober- oder unterhalb des Sliders angeklickt wurde. Je nach Mausposition wird dann um eine Seite nach unten bzw. oben geblättert. Am einfachsten ist der Klick auf einen der Pfeile, der ja nur den Verschub um eine Zeile nach oben oder bewirkt.

Wird der Dialog auf anderem Wege verlassen, wird auch die innere Schleife beendet. In der äußeren Schleife erfolgt die Entfernung der Dialogbox und ggf. eine Verzweigung in eine der anderen Funktionen (Hilfe, Parameter laden/speichern). Ist auch hier keine andere Funktion mehr gewünscht, also "OK" oder "Abbruch" gewählt worden, wird auch die äußere Schleife verlassen. Die Wahl von "Abbruch" bewirkt noch, daß die alten Einstellungen zurückgesetzt werden.

Die nächste Prozedur - *HandleAcc* - wartet auf einen *MessageEvent* und ruft die Dialogverwaltung auf, falls unser Accessory über das Menü angewählt wurde. Im Falle, daß ein Fremdaufruf erfolgt ist, antwortet Wodan darauf. Die dazugehörige Prozedur wird dann in der nächsten und letzten Folge besprochen.

```
 1: (************************************************************)
 2: (* Modulname              : HandleWodan (IMPLEMENTATION)    *)
 3: (* Copyright              : MAXON Computer GmbH             *)
 4: (* Datum                  :  1. Juni 1989                   *)
 5: (* letztes Edier-Datum    :  1. September 1989              *)
 6: (* Version                : 1.00b                           *)
 7: (* Entwicklungssystem     : Megamax Modula-2                *)
 8: (************************************************************)
 9:
10: IMPLEMENTATION MODULE HandleWodan;
11: (*$Q+,M-,N-,V+,P-,R-,S-*)
12:
13:
14: (* Resource-Datei *)
15:                                 IMPORT Wodan;
16:
17: (* allgemeine GEM-Deklarationen *)
18: FROM GEMGlobals                 IMPORT PtrObjTree, Root,
                                           ObjState,
                                           MaxDepth,
19:                                        MouseButton,
                                           MButtonSet,
                                           SpecialKeySet,
20:                                        PtrMaxStr;
21: FROM GEMEnv                     IMPORT InitGem, RC,
                                           DeviceHandle,
                                           GemError;
22: FROM GEMUtility                 IMPORT ShowBusy,
                                           ShowArrow,
                                           ShowMouse,
                                           HideMouse,
23:                                        objectSpace,
                                           objOffsetSpace,
                                           clearObjState,
24:                                        setObjState,
                                           getTextString,
                                           setTextString,
25:                                        setTextChar,
                                           prepareBox,
                                           releaseBox;
26:
27: (* AES-Importe *)
28: FROM AESEvents                  IMPORT accOpen,accClose,MessageBuffer,
                                           MessageEvent,
29:                                        TimerEvent;
30: FROM AESForms                   IMPORT FormAlert,
                                           FormDo;
31: FROM AESGraphics                IMPORT SlideBox,SlideDirection,
                                           MouseKeyState;
32: FROM AESMenus                   IMPORT RegisterAcc;
33: FROM AESMisc                    IMPORT SelectFile;
34: FROM AESObjects                 IMPORT DrawObject;
35: FROM AESWindows                 IMPORT MouseControl,
                                           UpdateWindow;
36: FROM AESResources               IMPORT LoadResource,FreeResource,
                                           ResourceAddr,
37:                                        ResourcePart;
38:
39: FROM ObjHandler                 IMPORT ObjectState;
40:
41: (* allgemeine Routinen *)
42: FROM GrafBase                   IMPORT Point, Rectangle;
43: FROM Keyboard                   IMPORT SpecialCode;
44: FROM Strings                    IMPORT String, Concat,StrEqual, Empty;
45: FROM CommunicationWodan         IMPORT ForeignCall,SendToAppl;
46:
47: (* Datei-Routinen *)
48: FROM Files                      IMPORT Create, Open,Close, Access,
                                           File, State,
49:                                        ResetState,ReplaceMode;
50: FROM Binary                     IMPORT WriteBlock,ReadBlock;
51:
52: FROM Directory                  IMPORT DriveToStr,DefaultDrive,SplitPath;
53:
54: (* System-Routinen *)
55: FROM SysUtil0                   IMPORT InWord, WExcl;
56: FROM SYSTEM                     IMPORT ADDRESS, ADR;
57:
58:
59: (* Konstanten *)
60: CONST ProgVersion      = '1.00'; (* Programm-Version *)
61:       ParameterHeader  = 'heop'; (* Header der Parameter-Datei *)
62:       ACCName          = "  Wodan..."; (* Name des Accessorys im Menü*)
63:       ResourceName     = "WODAN.RSC"; (* Name der Resource-Datei *)
64:       ParameterName    = "WODAN.INF";
    (* Name der Standard-Parameter-Datei   *)                            →
```

Es bleibt nur noch *CancelAcc*. Diese Cancel-Routine muß bei einem Accessory anders aussehen, da ein Accessory im Gegensatz zu "normalen" Programmen nicht verlassen werden darf. Es käme sonst zu einem Absturz, der ja unter allen Umständen vermieden werden soll.

Die Verwaltung des Sliders erfolgt bei Wodan explizit über mit Namen versehene Objekte. Dies ist jedoch bei Programmen, die mehrere Slider besitzen und daher eine universellere Funktion benötigen, sehr störend. Um eine universellere Funktion zu gestalten, könnte man einer Prozedur lediglich den Namen des Parent-Objekts übergeben. Diese Prozedur müßte dann nur die neue Stellung des Sliders zurückgeben. Die Unterscheidung der einzelnen Objekte des kompletten Sliders könnte mit Hilfe der erweiterten Objekttypen erfolgen. Alle bekannten Resource-Construction-Programme verfügen über die Fähigkeit, diese erweiterten Typen zu benutzen. "Erweitert" ist in dem Sinne zu verstehen, daß die bekannten Objekttypen innerhalb der Objektstruktur zwar ein Word (2 Byte) belegen, jedoch nur ein Byte (das untere) benötigen. Der erweiterte Objekttyp hat nun auch einen Wert im oberen Byte, der individuell verwendet werden darf. Man könnte also einen erweiterten Typ 1 für den Pfeil nach oben, eine 2 für den Pfeil nach unten und folgende Typen für die restlichen im Slider vorkommenden Objekte benutzen.

Nun Schluß mit den Erklärungen und den Anregungen. Sie möchten sicherlich beginnen, das Listing einzutippen. Viel Spaß dabei!

*Bis zum nächsten Mal...*

*Dietmar Rabich*

**Literatur:**


[1] ATARI ST Profibuch, H.-D. Jankowski/ J. F. Reschke/ D. Rabich, 6. Auflage, Sybex-Verlag 1988/89, S. 467ff, 495ff

[2] GEM Programmier-Handbuch, P. Balma/ W. Fitler, Sybex-Verlag 1988, S. 35ff


```
 65:        FSelMask         = "\WODAN*.INF";
       (* Maske für nachladbare Par.-Dateien *)
 66:        NoResource       = "[3][ Resourcefile
                               |WODAN.RSC| fehlt!| ][ OK ]";
 67:        CancelString     = "[3][ WODAN kann nicht|
                       installiert werden. | ][ OK ]";
 68:
 69:
 70:  (* Typ *)
 71:  TYPE  ParameterStruct = RECORD (* für Parameter-Dateien *)
 72:                            AltAktiv    : BOOLEAN;(* Wodan an/aus    *)
 73:                            AltPosition :SpecialCode;
                        (* Slider-Position *)
 74:                            AltStrings  :StringArray (* Texte *)
 75:                          END;
 76:
 77:
 78:  (* Variablen *)
 79:  VAR AccName, ParName, ParPath        : String;
 80:      MenuID, VoidCard                 : CARDINAL;
 81:      Device                           : DeviceHandle;
 82:      Kennung                          : ARRAY [alt1..altZ] OF CHAR;
 83:      LokalParaBlock                   : LokalParameter;
 84:      AuswahlBox, HilfeBox             : PtrObjTree;(* Dialog-Boxen   *)
 85:      DiskDefekt, NoInstal, StandPar   : PtrMaxStr;(* Alert-Boxen    *)
 86:
 87:
 88:  (* Warnmeldung "Keine Installation möglich." *)
 89:  PROCEDURE NoInstallation;
 90:
 91:    BEGIN
 92:      FormAlert (1, NoInstal^, VoidCard)
 93:    END NoInstallation;
 94:
 95:
 96:  (* Stringadresse setzen *)
 97:  PROCEDURE SetStringAdr (SAdr : ADDRESS);
 98:
 99:    BEGIN
100:      LokalParaBlock.StringAdr := SAdr
101:    END SetStringAdr;
102:
103:
104:  (* Initialisierung des Accessorys *)
105:  PROCEDURE InitAcc : BOOLEAN;
106:
107:    VAR success : BOOLEAN;
108:
109:    BEGIN
110:
111:      InitGem (RC, Device, success);(* GEM initialisieren *)
112:      IF ~success
113:      THEN
114:        RETURN FALSE
115:      END;
116:
117:      AccName := ACCName;              (* Accessoryanmelden *)
118:      RegisterAcc (ADR(AccName), MenuID, success);
119:      IF ~success
120:      THEN
121:        RETURN FALSE
122:      END;
123:
124:      LoadResource (ResourceName);(* Resource-Datei laden *)
125:      IF GemError ()
126:      THEN
127:       FormAlert (1, NoResource, VoidCard);
128:       RETURN FALSE
129:      END;
130:
131:      AuswahlBox := ResourceAddr (treeRsrc,Auswahl); (*Resource-Datei*)
132:      HilfeBox   := ResourceAddr (treeRsrc,Hilfe);   (* auswerten     *)
133:
134:      DiskDefekt := ResourceAddr (textString,Dierror);
135:      NoInstal   := ResourceAddr (textString,Noinstal);
136:      StandPar   := ResourceAddr (textString,Standard);
137:
138:      Kennung [alt0] := '0';    Kennung [alt1] :='1'; (* Tastenkürzel*)
139:      Kennung [alt2] := '2';    Kennung [alt3] :='3'; (* setzen      *)
140:      Kennung [alt4] := '4';    Kennung [alt5] :='5';
141:      Kennung [alt6] := '6';    Kennung [alt7] :='7';
142:      Kennung [alt8] := '8';    Kennung [alt9] :='9';
143:      Kennung [altA] := 'A';    Kennung [altB] :='B';
144:      Kennung [altC] := 'C';    Kennung [altD] :='D';
145:      Kennung [altE] := 'E';    Kennung [altF] :='F';
146:      Kennung [altG] := 'G';    Kennung [altH] :='H';
```

→

```
289:          LokalParaBlock.StringAdr^[i] :=
                                 Parameter.AltStrings[i]
290:        END;
291:        LokalParaBlock.Aktiv      :=
                                 Parameter.AltAktiv;
292:        LokalParaBlock.Position :=
                                 Parameter.AltPosition;
293:
294:        ShowArrow
295:      END
296:    END LoadParameter;
297:
298:
299:  (* Parameter speichern *)
300:  PROCEDURE SaveParameter ;
301:
302:    VAR OKButton, Success : BOOLEAN;
303:        Path, Name        : String;
304:        i                 : SpecialCode;
305:        IORes             : INTEGER;
306:        Parameter         : ParameterStruct;
307:
308:    BEGIN
309:      UpdateWindow (TRUE);
310:      SelectFile (ParPath, ParName, OKButton);
            (* File-Selector        *)
311:      UpdateWindow (FALSE);
312:
313:      IF OKButton AND
314:         ~Empty(ParName)
315:      THEN
316:        ShowBusy;
317:        SplitPath (ParPath, Path, Name);
            (* Pfad mit Dateinamen *)
318:        Concat (Path, ParName, Name, Success);
319:
320:        FOR i := alt1 TO altZ
321:        DO
322:          Parameter.AltStrings[i] :=
                             LokalParaBlock.StringAdr^[i]
323:        END;
324:        Parameter.AltAktiv      :=
             LokalParaBlock.Aktiv;
325:        Parameter.AltPosition :=
                             LokalParaBlock.Position;
326:        WriteParameter (Name, Parameter, IORes);
              (* Parameter schreiben *)
327:
328:        ShowArrow
329:      END
330:    END SaveParameter;
331:
332:
333:  (* erste Initialisierung *)
334:  PROCEDURE InitParameter : BOOLEAN;
335:
336:    VAR i         : SpecialCode;
337:        IORes     : INTEGER;
338:        Parameter : ParameterStruct;
339:
340:    BEGIN
341:
342:      (* Suche mit ShellFind kann hinzugefügt
            werden! *)
343:      ReadParameter (ParameterName, Parameter,
                         IORes); (* Parameter lesen *)
344:
345:      FOR i := alt1 TO altZ
346:      DO
347:        LokalParaBlock.StringAdr^[i] :=
                                 Parameter.AltStrings[i]
348:      END;
349:      LokalParaBlock.Aktiv      :=
                                 Parameter.AltAktiv;
350:      LokalParaBlock.Position :=
                                 Parameter.AltPosition;
351:
352:      RETURN LokalParaBlock.Aktiv
353:    END InitParameter;
354:
355:
356:  (* Hauptdialog durchführen *)
357:  PROCEDURE HandleStrings : BOOLEAN;
358:
359:    VAR MemString     : StringArray;
360:        i, MemPos     : SpecialCode;
361:        ReturnButton,
362:        NPos          : CARDINAL;
363:        space         : Rectangle;
364:        DoubleClick,
365:        MercAktiv     : BOOLEAN;
366:        MousePos      : Point;
367:        Buts          : MButtonSet;
368:        Keys          : SpecialKeySet;
369:        lSpace        : Rectangle;
370:
371:
372:    (* nachfolgende Position *)
373:    PROCEDURE IncSC (Special : SpecialCode; i :
                   CARDINAL) : SpecialCode;
374:
375:      BEGIN
376:        INC (Special, i);
377:        RETURN Special
378:      END IncSC;
379:
380:
381:    (* Hilfeseite ausgeben *)
382:    PROCEDURE DoHelp;
383:
384:      VAR ReturnButton : CARDINAL;
385:          space        : Rectangle;
386:          HString      : String;
387:
388:      BEGIN
389:        HString := ProgVersion;
390:        setTextString (HilfeBox, Version, HString);
391:        space := prepareBox (HilfeBox);
392:        FormDo (HilfeBox, Root, ReturnButton);
393:        clearObjState (ReturnButton, selectObj,
                           TRUE);
394:        releaseBox (HilfeBox, space)
395:      END DoHelp;
396:
397:
398:    (* Slider setzen *)
399:    PROCEDURE DrawSlider;
400:
401:      BEGIN
402:        AuswahlBox^[Slider].space.y :=
              (INTEGER(ORD(LokalParaBlock.Position) -
403:                                      ORD(alt1))
                                              *
404:              (AuswahlBox^[Slback].space.h         -
405:         AuswahlBox^[Slider].space.h))        DIV 26;
406:        DrawObject (AuswahlBox, Slback, MaxDepth,
                    objOffsetSpace(Cslider))
407:      END DrawSlider;
408:
409:
410:    (* Strings ausgeben *)
411:    PROCEDURE OutStrings (Draw : BOOLEAN);
412:
413:      BEGIN
414:        HideMouse;
415:        WITH LokalParaBlock
416:        DO
417:          setTextString (AuswahlBox, Text1,
418: LokalParaBlock.StringAdr^[       Position    ]);
419:          setTextString (AuswahlBox, Text2,
420: LokalParaBlock.StringAdr^[IncSC (Position,1)]);
421:          setTextString (AuswahlBox, Text3,
422: LokalParaBlock.StringAdr^[IncSC (Position,2)]);
423:          setTextString (AuswahlBox, Text4,
424: LokalParaBlock.StringAdr^[IncSC (Position,3)]);
425:          setTextString (AuswahlBox, Text5,
426: LokalParaBlock.StringAdr^[IncSC (Position,4)]);
427:          setTextString (AuswahlBox, Text6,
428: LokalParaBlock.StringAdr^[IncSC (Position,5)]);
429:          setTextString (AuswahlBox, Text7,
430: LokalParaBlock.StringAdr^[IncSC (Position,6)]);
431:          setTextString (AuswahlBox, Text8,
432: LokalParaBlock.StringAdr^[IncSC (Position,7)]);
433:          setTextString (AuswahlBox, Text9,
434: LokalParaBlock.StringAdr^[IncSC (Position,8)]);
435:          setTextString (AuswahlBox, Text10,
436: LokalParaBlock.StringAdr^[IncSC (Position,9)]);
437:          setTextChar   (AuswahlBox, Char1,
              Kennung   [        Position    ]);
438:          setTextChar   (AuswahlBox, Char2,
              Kennung   [IncSC (Position,1)]);
```

→

Hurra,

wir sind die

Nr. 1

ATARI® DeskTop Publishing

ATARI DeskTop Publishing System

Sie können aber auch für weniger Leistung mehr Geld ausgeben.

Das sind einige Leistungsbeispiele:

Graphiken und Bilder einsetzen und einscannen.

Gestalten: Wie es Ihnen gefällt. Sie können Ihren Text aus anderen Systemen übernehmen, einspaltig plazieren oder auch in noch mehr Textspalten aufteilen. Mit dem ATARI DTP System.

Graphische Grundelemente:

Formatierungsarten: Textaufbau Blocksatz Linksbündiger Aufbau Rechtsbündiger Aufbau zentrierter Textfluß Mittelachse

Text um ein Bild formatieren: Ein weiteres Beispiel für die Vielseitigkeit des ATARI DTP Systems: mit diesem können Sie jeden beliebigen Text um eine Darstellung formatieren. ATARI DTP Systeme.

Schriftgrößen

Drehtext Drehtext Drehtext

Unterschneidung von Schriften

Sonderzeichen: $ * ë ï â Ç Æ ¥ æ ¿ Ñ Œ Ø ...und eigene Kreationen.

Verschiedene Schriftstile: Underlined Shadow Outlined Superscript und Subscript

Damit macht Gestalten noch mehr Spaß, weil Leistung und Preis stimmen.

ATARI hat ein DeskTop Publishing System entwickelt, das Maßstäbe setzt.

Ob Sie Ihre Produkt- oder Angebotsblätter, Prospekte, Plakate, Zeitung, Bücher oder was auch immer gestalten wollen. Mit dem ATARI DeskTop Publishing System nutzen Sie echte Spitzentechnologie. Schnell haben Sie sich eingearbeitet in das professionelle Softwareprogramm CALAMUS.

Erleben Sie selbst, zu welchen Leistungen Sie fähig sind. Nehmen Sie uns beim Wort. In einem der ATARI DeskTop Publishing Center.

| Das ATARI DeskTop Publishing System | | |
|---|---|---|
| Computer | MEGA ST 2/4 | 2 MB RAM/4 MB RAM |
| Festplatte | MEGAFILE 30/60 | 30 MB/60 MB |
| Monitor | SM 124/SM 194 | 12 Zoll hochauflösend/19 Zoll Großbildschirm |
| Laserdrucker | SLM 804 | Direktmodus, gestochen scharf, 8 Seiten pro Minute |




...wir machen Spitzentechnologie preiswert.

```
147:      Kennung [altI] := 'I';    Kennung [altJ] :=
          'J';
148:      Kennung [altK] := 'K';    Kennung [altL] :=
          'L';
149:      Kennung [altM] := 'M';    Kennung [altN] :=
          'N';
150:      Kennung [altO] := 'O';    Kennung [altP] :=
          'P';
151:      Kennung [altQ] := 'Q';    Kennung [altR] :=
          'R';
152:      Kennung [altS] := 'S';    Kennung [altT] :=
          'T';
153:      Kennung [altU] := 'U';    Kennung [altV] :=
          'V';
154:      Kennung [altW] := 'W';    Kennung [altX] :=
          'X';
155:      Kennung [altY] := 'Y';    Kennung [altZ] :=
          'Z';
156:
157:      ParName := '';
    (* Name und Pfad der Parameter-Dateien *)
158:      Concat (DriveToStr (DefaultDrive ()),
              FSelMask, ParPath, success);
159:
160:      RETURN TRUE
161:
162:    END InitAcc;
163:
164:
165:  (* Disk-Fehlermeldung *)
166:  PROCEDURE DiskError (errNumber : INTEGER);
167:
168:    BEGIN
169:  (* Fehlernummer wird hier nicht ausgewertet *)
170:      FormAlert (1, DiskDefekt^, VoidCard)
171:    END DiskError;
172:
173:
174:  (* Fehler-Statur ermitteln und zurücksetzen *)
175:  PROCEDURE errState (f : File; VAR IOResult :
              INTEGER) : BOOLEAN;
176:
177:    BEGIN
178:      IOResult := State(f);
179:      IF IOResult < 0
180:      THEN
181:        ResetState (f);
182:        RETURN TRUE
183:      END;
184:      RETURN FALSE
185:    END errState;
186:
187:
188:  (* Parameter aus Datei lesen *)
189:  PROCEDURE ReadParameter (FileName      : ARRAY OF
              CHAR;
190:                       VAR Paras      :
                                ParameterStruct;
191:                       VAR IOResult  : INTEGER);
192:
193:    VAR OptFile    : File;
194:        HArray,
195:        OptHeader  : ARRAY [0..3] OF CHAR;
196:
197:
198:      (* Standard-Parameter setzen *)
199:      PROCEDURE StandardParameter;
200:
201:        VAR i : SpecialCode;
202:
203:        BEGIN
204:          WITH Paras
205:          DO
206:            AltAktiv    := FALSE;  (* Wodan passiv*)
207:            AltPosition := alt1;   (* Slider am
                                        oberen Ende *)
208:            FOR i := alt1 TO altZ (*Keine Strings*)
209:            DO
210:              AltStrings[i] := ''
211:            END
212:          END;
213:
214:          FormAlert (1, StandPar^, VoidCard)
215:        END StandardParameter;
216:
217:
218:    BEGIN
219:      Open (OptFile, FileName, readOnly);
          (* Datei öffnen          *)
220:      IF errState (OptFile, IOResult)
          (* Fehler aufgetreten? *)
221:      THEN
222:        DiskError (IOResult);
223:        StandardParameter
224:      ELSE
225:        ReadBlock (OptFile, OptHeader);
           (* Header lesen          *)
226:        HArray := ParameterHeader;
227:        IF errState (OptFile, IOResult) OR
           (* Fehler? Header OK?  *)
228:          ~StrEqual (OptHeader, HArray)
229:        THEN
230:          StandardParameter
231:        ELSE
232:          ReadBlock (OptFile, Paras);
             (* Parameter lesen      *)
233:          IF errState (OptFile, IOResult)
234:          THEN
235:            StandardParameter
236:          END
237:        END
238:      END;
239:      Close (OptFile)    (* Datei schliePen *)
240:    END ReadParameter;
241:
242:
243:  (* Parameter in Datei schreiben *)
244:  PROCEDURE WriteParameter (FileName        : ARRAY
                                            OF CHAR;
245:                            VAR Paras     :
                                     ParameterStruct;
246:                            VAR IOResult :INTEGER);
247:
248:    VAR OptFile : File;
249:        HArray  : ARRAY [0..3] OF CHAR;
250:
251:    BEGIN
252:      Create (OptFile, FileName, writeOnly,
                  replaceOld); (* Datei anlegen *)
253:      IF errState (OptFile, IOResult)
                            (* Fehler?          *)
254:      THEN
255:        DiskError (IOResult)
256:      ELSE
257:        HArray := ParameterHeader;
258:        WriteBlock (OptFile, HArray);
             (* Header und           *)
259:        WriteBlock (OptFile, Paras)
             (* Parameter schreiben *)
260:      END;
261:      Close (OptFile)
             (* Datei schliePen       *)
262:    END WriteParameter;
263:
264:
265:        (* Parameter laden *)
266:  PROCEDURE LoadParameter ;
267:
268:    VAR OKButton, Success  : BOOLEAN;
269:        Path, Name         : String;
270:        i                  : SpecialCode;
271:        IORes              : INTEGER;
272:        Parameter          : ParameterStruct;
273:
274:    BEGIN
275:      UpdateWindow (TRUE);
276:      SelectFile (ParPath, ParName, OKButton);
            (* File-Selector         *)
277:      UpdateWindow (FALSE);
278:
279:      IF OKButton AND
280:         ~Empty(ParName)
281:      THEN
282:        ShowBusy;
283:        SplitPath (ParPath, Path, Name);
             (* Pfad mit Dateinamen *)
284:        Concat (Path, ParName, Name, Success);
285:
286:        ReadParameter (Name, Parameter, IORes);
                       (* Parameter lesen       *)
287:        FOR i := alt1 TO altZ
288:        DO
```

→

## Spiele

9

| | | | | | |
|---|---|---|---|---|---|
| Armada | 94,- * | Full Metal Planet | 84,- * | Rings of Medusa | 84,- * |
| Balance of Power 1990's | 79,- * | G.NIUS | 64,- * | Roger Rabbit | 74,- * |
| Ballistix | 64,- * | Great Courts | 79,- * | RVF Honda | 79,- * |
| Battletech | 79,- * | Gunship | 79,- * | Snoopy | 64,- * |
| Bloodwych | 79,- * | Hillsfar (AD&D) | 79,- * | Space Quest III | 74,- * |
| Borodino | 89,- * | Iron Lord | 89,- * | Starglider II (f + s/w) | 74,- * |
| Castle Warrior | 64,- * | Jeanne d'Arc | 59,- * | Stuntcar Racer | 84,- * |
| Chicago '90 | 64,- * | Kaiser | 119,- * | Super Quintet | 69,- * |
| Das Reich 1871 | 54,- * | Kult | 67,- * | Table Tennis | 64,- * |
| Dungeon Master Exp. Set #1 | 64,- * | Legend of Djel | 59,- * | TOOBIN' | 64,- * |
| DR. Lustig | | Licence to kill | 59,- * | Total Eclipse | 79,- * |
| Der Spezialist für Vornamen | | Manhunter 2 | 89,- * | Triad 2 | 79,- * |
| und Persönlichkeitsananlyse. | 29,- * | Nevermind | 64,- * | Turbo Out-Run | 64,- * |
| Elite | 79,- * | North & South | 79,- * | TV-Sport Football | 84,- * |
| Esprit | 99,- * | Oil Imperium | 59,- * | Ultima V | 94,- * |
| E.S.S. | 94,- * | Omega | 94,- * | Wall Street Wizard (f + s/w) | 64,- * |
| F-16 Falcon | 84,- * | Pharao | 84,- * | Waterloo 1815 | 74,- * |
| F-16 Mission Disk | 64,- * | Pirates | 79,- * | Weird Dreams | 72,- * |
| Fighter Bomber | 79,- * | Populous | 79,- * | West Phaser | 119,- * |
| Flight Simulator 2 (f + s/w) | 119,- * | Populous, Promised Land | 39,- * | XENON 2 - Megablast | 74,- * |

**CADjA V.1.2** **998,- *** 10

CAD-Programm der Spitzenklasse, anwendbar für Elektronik, Maschinenbau, Architektur. Plotausgabe bis DIN A0. Symbolbibliotheken verfügbar. ASCII- Schnittstelle. Direkte Weiterverarbeitung mit "CADjA-CAM". Schnittstelle zu STEVE. Leistungsmerkmale: überragend! Druckausgabe auf Nadeldrucker, Laser, Plotter. Demo: DM 50,- *

**STar Designer** **DM 149,- ***

**ARABESQUE** **DM 278,- ***

**VOMBLATT** **DM 89,- *** 12

Musikdidaktisches Programm zum Erlernen des Notenlesens - auch ohne Vorkenntnisse - bis Hochschulniveau. Alle Schlüssel, Ein-/Ausgabe über MIDI möglich, Prüffunktionen und mehr...

**Steinberg 'Twelve'** **DM 99,- ***

12 Spur Midi Sequencer. 'Twelve' ist der kleine Bruder des schon populär gewordenen Steinberg Twenty-Four.

**Cubase** **DM 790,- ***

Desktop Midi Recording System

**CALAMUS 1.09** **DM 748,- *** 11

DTP-Programm für den professionellen Einsatz. Diese Anzeige wurde komplett mit Calamus erstellt und auf der Linotronic 300 ausbelichtet.

**OUTLINE ART** **DM 398,- ***

Vektor-Art-Programm für CALAMUS. Freiraum für kreatives Gestalten von Schrift und Grafik.

**PKS-Write** **DM 198,- ***

Der Texteditor für CALAMUS ist da!

**DTP-Service** **a. A.**

Schulung, Layout, Produktion, Scan-Service ...

**BOOT-IT** **DM 69,- *** 18

...und Booten von der Festplatte macht Spaß!

+ GEM-Anwendungen starten automatisch
+ ACCs und TOS (im AUTO-Ordner) wählbar
+ BATCH-Dateien mit - Accessories
  - TOS Anwendungen - Stan.-Zugriffspfad
  - Namen der GEM-Anw. - Blitter ein/aus
  - MED- oder LO-Res Einstellung
+ Korrektes Desktop für jede Auflösung
+ Update-Service, Handbuch und Tel. Hot-Line!

**CALAMUS-FONTS** **ab DM 59,- *** 11

AKTIVA | AltBerlin bold | GEODET | Bobble + Plup | INFRA | JILLY | Boedet | Peking reg | BONUM | ROCA + YAPPIE | CARDPLAY | ROMA REG | Caslo + Skript | Rund

**Desk Assist 4.2** **DM 188,- *** 19

Das Multifunktions-Accessory! Unverzichtbar für jeden Atari-Anwender. Terminplaner mit Alarm- und Dauerterminen, Adressen- und Telefondatei für Serienbriefe, Druckerspooler, Taschenrechner (dezimal/hex/binär/Zeit/Datum) 18-stellige Genauigkeit, Maßumrechnung, Diskettenformatierung bis 830 KB, RAM und Diskeditor, ASCII-Tabelle, usw.

**Mortimer** **DM 79,- ***

**Harlekin** **DM 129,- ***

**Architektur & Design** **DM 59,- *** 11

Über 400 Vektor- und Rastergrafiken (PAC-Format) für CALAMUS und Mal-/Zeichenprogramme. Beispiele finden Sie auf diesen Seiten.

**Calamus Layout-Paket DM 79,-**

Gestaltungshilfe, Pass- und Schnittmarken in DIN Standardformaten A5, A4 und A3, hoch u. quer. Mit Anleitung (auf Wunsch in Englisch).

**RAM-Erweiterung 2,5 MB** **a. A.** 20

4 MB Speicherkarte (steckbar) von Weide, mit 2,5 MB bestückt. Einfacher Einbau, ohne Löten. Kein Bildflimmern, keine zusätzliche Software, Hardware oder Stromversorgung notwendig.

**RAM-Erweiterung 4 MB** **a. A.**

Wie oben, jedoch mit 4 MB bestückt.

**MiniRAM** 1MB-Erweiterung **DM 298,- ***

4-Bit organisierte Erweiterung für 260/520 ST. Nur 90x37mm groß! Selbsteinbau ohne Löten.

**Marconi Trackerball** **DM 198,- *** 11

100%tiger Mausersatz mit hoher Lebensdauer. Seit Jahren im Radar-/Flugsicherungseinsatz erprobt. Im DTP- und CAD-Bereich wegen des schnellen und präzisen Ansteuerns/Positionierens unentbehrlich. Wer ihn kennt, kann nicht mehr ohne ...

**Handyscanner Typ 10** **DM 898,- ***

400 dpi, 105mm Scannbreite, Texterkennung und Bildverarbeitungsprogramm.

## Kabel & Zubehör

20

| | |
|---|---|
| Joystick-/Maus Verlängerungskabel | 7,90 * |
| Druckerkabel Centronics montiert 2m | 15,90 * |
| Druckerkabel Centronics vergossen 2m | 18,90 * |
| Druckerkabel Centronics vergossen 3m | 23,00 * |
| Druckerkabel Centronics vergossen 5m | 32,90 * |
| Centronics-Interface (NEC empfohlen) | 84,90 * |
| Scart-Kabel 2m | 24,90 * |
| Floppy-Verlängerungskabel 2m | 28,90 * |
| Midi-Kabel 1,2m | 7,90 * |
| Midi-Kabel 5m | 14,50 * |
| RS 232/v 24 Kabel 2m | 12,90 * |
| Verlängerungskabel Festplatte 1m | 49,90 * |
| Mausmatte | 11,90 * |
| Disketten-Parkplatz (Befestigung am Rechner) | 6,90 * |
| Ergostick (ergonomischer Joystick) | 69,90 * |
| Quickjoy Jet Fighter | 29,90 * |

**Digital Image**
Postfach 1206
6096 Raunheim a.M.
06142 / 22636 od 43560

**IDL Software**
Lagerstraße 11
6100 Darmstadt 13
06151 / 58912

**RHD**
Am steinigen Berg 1
6101 Roßdorf
06154 / 8782

**KREATIV-Software**
Oberwürzbacher Str. 10
6676 Mandelbachtal
06803/3850 u. 06805/2666

**Musikinstrumente&Computer**
August-Bebelstr. 3
6840 Lampertheim 5
06241 / 80899

**Weeske Computer**
Potsdamer Ring 10
7150 Backnang
07191 / 1528-29 od. 60076

**Duffner's PD-Center**
Ritterstr. 6
7833 Endingen a.K.
07642 / 3875 od. 3739

**HAROSOFT**
Tomerdinger Straße 23
7909 Dornstadt
07348 / 22312 (Fax: 22729)

**Soft+Hardware LAUTERBACH**
Josephsplatz 3
8000 München 40
089 / 2722377

**Dietmar Schramm**
Promberg 6
8122 Penzberg
08856 / 7287

**Schick EDV-Systeme**
Hauptstraße 32a
8542 Roth
09171 / 5058-59

**PR8-SOFT, Klaus-M. Pracht**
Otto-Hahn-Str. 10
8702 Estenfeld
09305 / 8211

* Alle Preise sind unverbindlich empfohlene Verkaufspreise

# Coupon

Ich bestelle:

Name, Vorname

Straße

Plz, Ort

☐ Scheck liegt bei
☐ Per NN (Nur Inland, zuzügl. DM 6,- NN-Gebühr)

```
439:        setTextChar   (AuswahlBox, Char3,
              Kennung   [IncSC (Position,2)]);
440:        setTextChar   (AuswahlBox, Char4,
              Kennung   [IncSC (Position,3)]);
441:        setTextChar   (AuswahlBox, Char5,
              Kennung   [IncSC (Position,4)]);
442:        setTextChar   (AuswahlBox, Char6,
              Kennung   [IncSC (Position,5)]);
443:        setTextChar   (AuswahlBox, Char7,
              Kennung   [IncSC (Position,6)]);
444:        setTextChar   (AuswahlBox, Char8,
              Kennung   [IncSC (Position,7)]);
445:        setTextChar   (AuswahlBox, Char9,
              Kennung   [IncSC (Position,8)]);
446:        setTextChar   (AuswahlBox, Char10,
              Kennung   [IncSC (Position,9)])
447:      END;
448:      IF Draw
449:      THEN
450:        DrawObject (AuswahlBox, Editfeld,
                  MaxDepth, objOffsetSpace
                  (Editfeld))
451:      END;
452:      ShowMouse
453:    END OutStrings;
454:
455:
456:  BEGIN
457:
458:    FOR i := alt1 TO altZ  (* alten
                        Stand merken *)
459:    DO
460:      MemString[i] :=
                  LokalParaBlock.StringAdr^[i];
461:    END;
462:    MercAktiv := LokalParaBlock.Aktiv;
463:    MemPos    := LokalParaBlock.Position;
464:
465:    REPEAT
466:
467:      (* Vorbereitungen *)
468:      OutStrings (FALSE);
469:
470:      IF LokalParaBlock.Aktiv
471:      THEN
472:        setObjState (Altan, selectObj, FALSE);
473:        clearObjState (Altaus, selectObj, FALSE)
474:      ELSE
475:        clearObjState (Altan, selectObj, FALSE);
476:        setObjState (Altaus, selectObj, FALSE)
477:      END;
478:
479:      AuswahlBox^[Slider].space.h :=
           (AuswahlBox^[Slback].space.h * 10)
            DIV 36;
480:      AuswahlBox^[Slider].space.y :=
             (INTEGER(ORD(LokalParaBlock.Position) -
481:                                ORD(alt1))
                                        *
482:             (AuswahlBox^[Slback].space.h       -
483:       AuswahlBox^[Slider].space.h))      DIV 26;
484:
485:      space := prepareBox (AuswahlBox);
486:
487:      (* Dialog durchführen (mit Slider-
           Bewegung) *)
488:      REPEAT
489:        FormDo (AuswahlBox, Root, ReturnButton);
             (* Dialog durchführen        *)
490:        DoubleClick := InWord(15, ReturnButton);
             (* oberstes Bit auswerten *)
491:        WExcl (ReturnButton, 15);
492:        clearObjState (ReturnButton, selectObj,
                         TRUE);
493:
494:        WITH LokalParaBlock
495:        DO
496:          getTextString (AuswahlBox, Text1,
497: LokalParaBlock.StringAdr^[       Position    ]);
498:          getTextString (AuswahlBox, Text2,
499: LokalParaBlock.StringAdr^[IncSC(Position,1)]);
500:          getTextString (AuswahlBox, Text3,
501: LokalParaBlock.StringAdr^[IncSC(Position,2)]);
502:          getTextString (AuswahlBox, Text4,
503: LokalParaBlock.StringAdr^[IncSC(Position,3)]);
504:          getTextString (AuswahlBox, Text5,
505: LokalParaBlock.StringAdr^[IncSC(Position,4)]);
506:          getTextString (AuswahlBox, Text6,
507: LokalParaBlock.StringAdr^[IncSC(Position,5)]);
508:          getTextString (AuswahlBox, Text7,
509: LokalParaBlock.StringAdr^[IncSC(Position,6)]);
510:          getTextString (AuswahlBox, Text8,
511: LokalParaBlock.StringAdr^[IncSC(Position,7)]);
512:          getTextString (AuswahlBox, Text9,
513: LokalParaBlock.StringAdr^[IncSC(Position,8)]);
514:          getTextString (AuswahlBox, Text10,
515: LokalParaBlock.StringAdr^[IncSC(Position,9)])
516:        END;
517:
518:        LokalParaBlock.Aktiv := selectObj IN
                        ObjectState (Altan);
519:
520:        CASE ReturnButton
521:        OF
522:
523:          (* Slider angeklickt? *)
524:          Slider : MouseKeyState (MousePos, Buts,
                   Keys); (* Maustaste fest- *)
525:                   IF msBut1 IN Buts
                            (* gehalten?      *)
526:                   THEN
527:                     MouseControl (TRUE);
528:
529:                     (* Slider verschieben *)
530:                     SlideBox (AuswahlBox,
                              Slback, Slider,
                              vertSlide, NPos);
531:                     LokalParaBlock.Position :=
                           SpecialCode (ORD(alt1) +
532:                     (NPos * 26 + 500) DIV 1000);
533:                     DrawSlider;
534:
535:                     OutStrings (TRUE);
536:                     MouseControl (FALSE)
537:                   END|
538:
539:          (* Slider-Hintergrund angeklickt? *)
540:          Slback : MouseKeyState (MousePos, Buts,
                   Keys); (* Mausposition *)
541:                   lSpace := objOffsetSpace
                            (Slider);
542:                   IF MousePos.y > lSpace.y +
                      lSpace.h   (* Slider
                                   nach unten *)
543:                   THEN
544:                     IF LokalParaBlock.Position
                        > altH
545:                     THEN
546:                       LokalParaBlock.Position
                          := altQ
547:                     ELSE
548:                       INC
                         (LokalParaBlock.Position, 9)
549:                     END
550:                   ELSIF MousePos.y < lSpace.y
                            (* Slider
                               nach oben  *)
551:                   THEN
552:                     IF LokalParaBlock.Position
                        < alt0
553:                     THEN
554:                       LokalParaBlock.Position
                          := alt1
555:                     ELSE
556:                       DEC
                         (LokalParaBlock.Position, 9)
557:                     END
558:                   END;
559:                   DrawSlider;
560:                   OutStrings (TRUE)|
561:
562:          (* Pfeil nach oben angeklickt? *)
563:          Up     : IF LokalParaBlock.Position >
                   alt1
564:                   THEN
565:                     IF DoubleClick
                            (* Doppelklick?     *)
566:                     THEN
567:                       LokalParaBlock.Position
                          := alt1 (* ganz
                          nach unten *)
568:                     ELSE
```

→

```
569:                  DEC
                        (LokalParaBlock.Position)
                        (* eine Position    *)
570:                END;    (* nach unten   *)
571:                DrawSlider;
572:                OutStrings (TRUE)
573:              END|
574:
575:        (* Pfeil nach unten angeklickt? *)
576:        Down    : IF LokalParaBlock.Position <
                      altQ
577:                  THEN
578:                    IF DoubleClick
                          (* Doppelklick?     *)
579:                    THEN
580:                      LokalParaBlock.Position
                            := altQ (* ganz
                                     nach oben   *)
581:                    ELSE
582:                      INC
                            (LokalParaBlock.Position)
                            (* eine Position    *)
583:                    END;
                              (* nach oben    *)
584:                    DrawSlider;
585:                    OutStrings (TRUE)
586:                  END
587:
588:      END
589:
590:    UNTIL (ReturnButton = Bok    ) OR
                        (* OK?                    *)
591:          (ReturnButton = Babbr ) OR
                        (* Abbruch?               *)
592:          (ReturnButton = Bhilfe) OR
                        (* Hilfe?                 *)
593:          (ReturnButton = Bload ) OR
                        (* Parameter laden?       *)
594:          (ReturnButton = Bsave );
                        (* Parameter speichern? *)
595:
596:    releaseBox (AuswahlBox, space);
597:
598:    CASE ReturnButton
599:    OF
600:      (* Hilfe? *)
601:      Bhilfe : DoHelp            |
602:
603:      (* Parameter laden? *)
604:      Bload  : LoadParameter     |
605:
606:      (* Parameter speichern? *)
607:      Bsave  : SaveParameter
608:
609:    END
610:
611:  UNTIL (ReturnButton = Bok)      OR
              (* OK?                 *)
612:        (ReturnButton = Babbr);
              (* Abbruch?            *)
613:
614:              (* Abbruch? *)
615:      IF ReturnButton = Babbr
616:      THEN
617:        FOR i := alt1 TO altZ
                (* Veränderungen rückgängig machen *)
618:        DO
619:          LokalParaBlock.StringAdr^[i] :=
                                          MemString[i]
620:        END;
621:        LokalParaBlock.Aktiv    := MercAktiv;
622:        LokalParaBlock.Position := MemPos
623:      END;
624:
625:      RETURN LokalParaBlock.Aktiv
626:
627:    END HandleStrings;
628:
629:
630:  (* Accessory wartet auf Aktivierung *)
631:  PROCEDURE HandleAcc : BOOLEAN;
632:
633:    VAR   MsgBuffer   : MessageBuffer;
634:          ReturnCode  : BOOLEAN;
635:
636:    BEGIN
637:      LOOP
638:        MessageEvent (MsgBuffer);
639:        IF MsgBuffer.msgType = accOpen
              (*                Accessory gewählt? *)
640:        THEN
641:          IF MsgBuffer.aOpnMId = MenuID
              (*                unser Accessory ?  *)
642:          THEN
643:            ReturnCode := HandleStrings ();
644:            EXIT
645:          END
646:        ELSIF MsgBuffer.msgType = ForeignCall
              (*                Fremdaufruf?       *)
647:        THEN
648:          SendToAppl (MsgBuffer,ADR(LokalParaBlock))
649:        END
650:      END;
651:      RETURN ReturnCode
652:    END HandleAcc;
653:
654:
655:  (* Warteschleife bei Fehler *)
656:  PROCEDURE CancelAcc;
657:
658:    BEGIN
659:      FormAlert (1, CancelString, VoidCard);
660:      WHILE TRUE DO
661:        TimerEvent (10000L)
662:      END
663:    END CancelAcc;
664:
665:
666:  END HandleWodan.
```

# Logische Schaltungen

## und deren Vereinfachung nach Quine-McCluskey Teil 1




**Vor einiger Zeit (Oktober 89) wurden in der ST Computer universelle Logikbausteine, sogenannte GALs, vorgestellt, mit welchen logische Funktionen auf einfache Weise implementiert werden können [1]. Bei anspruchsvollen Aufgaben können diese Logikfunktionen, und damit auch deren Realisierung als Logikschaltung, recht kompliziert werden, so daß unter Umständen mehrere GALs zu deren Verwirklichung erforderlich sind. Um die Anzahl der GALs gering zu halten, empfiehlt es sich, die Logikfunktionen nach Möglichkeit zu vereinfachen.**

Im ersten Teil dieses Artikels werden der Einsatz von Logikschaltungen und deren Darstellung als logische Funktion sowie die Grundlagen der logischen Algebra, auch boolesche Algebra genannt, behandelt. Im zweiten Teil wird das sich auf diese Grundlagen stützende Vereinfachungsverfahren von Quine-McCluskey besprochen, für welches das PASCAL-Programm LOGIMIN geschrieben wurde. LOGIMIN wird im dritten Teil der vorliegenden Artikelserie veröffentlicht.

### Das Einsatzgebiet von Logikschaltungen

Kombinatorische Logikschaltungen, auch einfach als logische Schaltungen oder als Schaltwerke bezeichnet, dienen in der Digitaltechnik dazu, in Abhängigkeit von bestimmten Eingangszuständen *Entscheidungen zu treffen*, indem genau festgelegte Ausgangszustände erzeugt werden. Eine solche *Entscheidung* könnte z.B. die Auswahl eines Mikroprogramms im Steuerwerk eines Prozessors sein. Die Entscheidung selbst hängt hierbei von dem in den Befehlsdecodierer eintreffenden Steuerwort ab. Wir wollen uns jedoch ein weniger komplexes Beispiel für eine Entscheiderschaltung vornehmen, welches etwas näher betrachtet werden soll: in einem Kraftwerk werde dieselbe Größe von drei Meßgeräten gemessen, und es soll beim Überschreiten eines gefährlichen Wertes ein Warnsignal ausgelöst werden.

Da man damit rechnen muß, daß eines der Meßgeräte ausfällt, soll eine Sicherheitseinrichtung betätigt werden, wenn mindestens zwei Meßgeräte einen gefährlichen Zustand anzeigen. Zur formalen Beschreibung dessen, was die Entscheiderschaltung zu tun hat, verwendet man vorteilhaft sogenannte *binäre Variablen*, welche zu Ehren des englischen Mathematikers George Boole auch als *boolesche Variablen* bezeichnet werden.

Im Unterschied zu herkömmlichen arithmetischen Variablen, welche im allgemeinen unendlich viele Werte annehmen können, stehen für boolesche Variablen nur zwei Werte, die wir *1 und 0* nennen wollen, zur Verfügung. Die beiden Elemente 1 und 0 sind dabei nicht als Zahlen zu verstehen, sondern als zwei logische Zustände, die man auch mit JA und NEIN oder auch als WAHR und FALSCH bezeichnen könnte. Immer dann, wenn zur Entscheidungsfindung einfache *Fallunterscheidungen* ausreichen, ist der Einsatz von booleschen Variablen zur Beschreibung des Problems geeignet.

In unserem Beispiel ordnet man z.B. dem ersten Meßgerät die Variable $X_1$ zu, welche bei Vorliegen eines unkritischen Meßwertes auf 0, bei Auftreten eines kritischen Wertes auf 1 steht. Analog hierzu werden für das zweite und dritte Meßgerät die Variablen $X_2$ und $X_3$ eingesetzt. Definiert man noch eine Ausgangsvariable Y, welche genau dann den Zustand 1 annehmen soll, wenn ein Warnsignal auszulösen ist, kann man den verbal erläuterten Sachverhalt der Warnanlage formal mit einer sogenannten *Wahrheits- oder Wertetabelle* beschreiben, welche in Bild 1 dargestellt ist.

| Zeile | $X_3$ | $X_2$ | $X_1$ | Y |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 |

***Bild 1: Wahrheitstabelle für die Warnanlage mit "Mindestens 2 aus 3"-Entscheidung***

In dieser Wahrheitstabelle werden alle möglichen Zustände der Variablen $X_1$, $X_2$ und $X_3$ sowie der zugehörige Ausgangswert Y aufgelistet. Man erkennt, daß Y stets dann 1 ist, wenn mindestens zwei der Variablen $X_1$, $X_2$ und $X_3$ auf 1 stehen. Zur besseren Orientierung sind alle Zeilen der Wahrheitstabelle von 0 bis 7 durchnumeriert. Bei genauem Hinsehen erkennt man, daß die 0/1-Kombinationen unter den Variablen $X_i$, i=1,2,3, den *Dualcodierungen* der zugehörigen *Zeilennummern* entsprechen. Im vorliegenden Beispiel gilt also

Zeilenzahl = $X_3 * 2^2 + X_2 * 2^1 + X_1 * 2^0$ (1)

Es ist sinnvoll, die Wahrheitstabelle stets in dieser geordneten Weise aufzubauen, obwohl dies nicht zwingend notwendig ist. Die Anzahl der möglichen Zeilen und damit der möglichen Belegungskombinationen für die Variablen $X_i$, $i=1,2,...,k$ beträgt stets $2^k$, in unserem Beispiel also $2^3=8$. Die Durchnumerierung der Zeilen beginnt bei 0 und endet bei $2^k-1$.

Formuliert man den Inhalt der Wahrheitstabelle nach Bild 1 in Worten, so kann man sagen :

Y ist genau dann 1, wenn

$X_3=0$ und $X_2=1$ und $X_1=1$ (Zeile 3) oder wenn
$X_3=1$ und $X_2=0$ und $X_1=1$ (Zeile 5) oder wenn
$X_3=1$ und $X_2=1$ und $X_1=0$ (Zeile 6) oder wenn
$X_3=1$ und $X_2=1$ und $X_1=1$ (Zeile 7) ist.

Diese Darstellung der Wahrheitstafel scheint zunächst eher umständlich als hilfreich. Sie bildet jedoch das Bindeglied zur sogenannten *booleschen Algebra*, welche im folgenden besprochen werden soll.

## Boolesche Algebra

Jede Art von Algebra hat zum Ziel, einen Formalismus bereitzustellen, mit welchem die Eigenschaften bzw. die Struktur der zu bearbeitenden Aufgabe besser sichtbar, und die Aufgabe selbst besser bearbeitbar wird. Der verwendete Formalismus hat dabei zwei Hauptaufgaben zu erfüllen. Zum einen möchte man mit möglichst *wenig Schreibarbeit* auskommen, weswegen *abkürzende Schreibweisen* eingeführt werden. Aus der arithmetischen Algebra ist z.B. bekannt, daß die vier Grundrechenarten Addition, Subtraktion, Multiplikation und Division mit den Symbolen +, -, * und / abgekürzt werden. Ebenso werden i.allg. kurze Variablennamen verwendet. Bei einer einfachen Haushaltsbilanz könnte z.B. die Menge des eingenommenen Geldes mit E, die Menge des ausgegebenen Geldes mit A und das verbleibende Restgeld mit R bezeichnet werden. Das Restgeld berechnet sich dann nach R = E - A, was zweifellos eine sehr kompakte und übersichtliche Form der Haushaltsrechnung darstellt.

Die zweite Hauptaufgabe einer Algebra besteht darin, *Rechenregeln* zur Verfügung zu stellen, um die algebraischen Ausdrücke manipulieren zu können. Auch die boolesche Algebra hat die Erfüllung der beiden genannten Hauptaufgaben zum Ziel. Abkürzungen hatten wir bis zu einem gewissen Grad beim Erstellen der Wertetabelle nach Bild 1 bereits eingeführt, wir wollen dies aber noch weiter treiben.

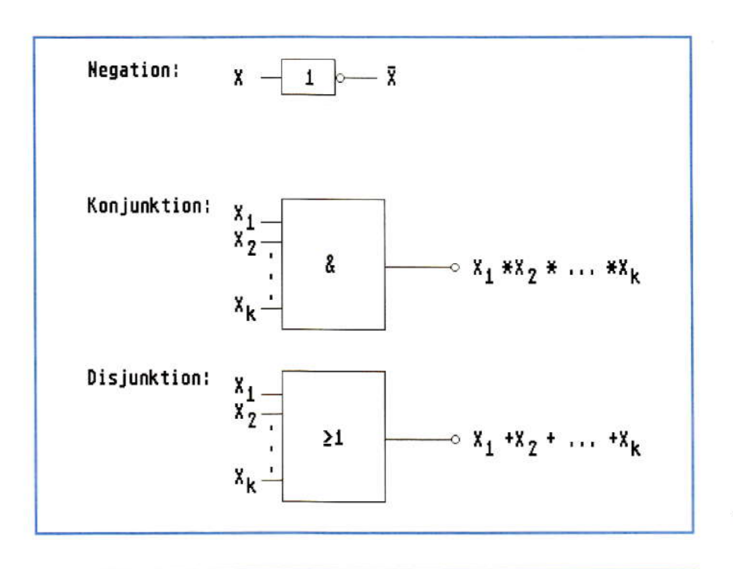



*Bild 2: Grafiksymbole für die logischen Grundverknüpfungen*

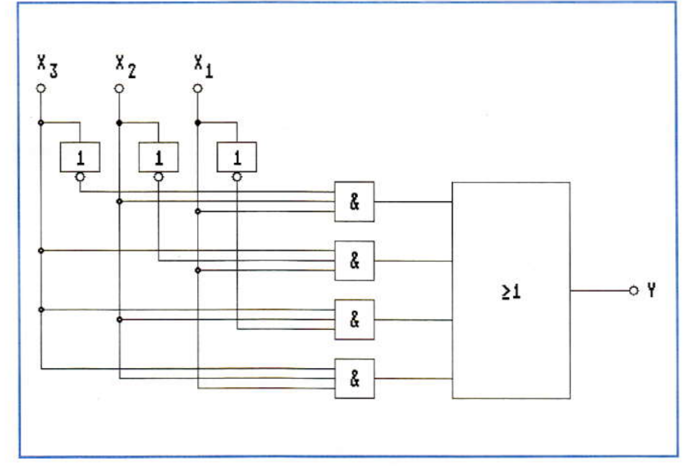



*Bild 3: Blockbilddarstellung von Glg. (2)*

Die *Grundoperationen* der booleschen Algebra sind:

die Nicht-Verknüpfung (Negation),
die Und-Verknüpfung (Konjunktion)
und die Oder-Verknüpfung (Disjunktion).

Wird eine boolesche Variable X *negiert*, so schreibt man dies in der Form $\overline{X}$ und meint damit

$\overline{X}=0$ falls $X=1$
bzw. $\overline{X}=1$ falls $X=0$.

Das grafische Symbol der Negation ist in Bild 2 dargestellt.

Die *Und-Verknüpfung* von booleschen Variablen $X_1$, $X_2$, ...,$X_k$ wird meist mit einem "Mal"-Zeichen abgekürzt, schreibt sich also

$X_1 * X_2 * ... * X_k$ oder auch einfach $X_1X_2 ... X_k$

und bedeutet, daß der gesamte Ausdruck, auch *Produktterm* genannt, genau dann 1 ist, falls $X_1$ und $X_2$ und ... und $X_k$ jeweils gleich 1 sind. Ansonsten ist der Ausdruck 0. Auch das Grafiksymbol der Konjunktion ist in Bild 2 zu sehen.

Die *Oder-Verknüpfung* von booleschen Variablen $X_1$, $X_2$, ..., $X_k$ wird meist mit einem "Plus"-Zeichen abgekürzt, schreibt sich damit

$X_1 + X_2 + ... + X_k$

und bedeutet, daß der gesamte Ausdruck genau dann 1 ist, falls $X_1$ oder $X_2$ oder ... oder $X_k$ oder auch mehrere bzw. alle der Variablen $X_1$ bis $X_k$ gleich 1 sind. Die grafische Darstellung der Disjunktion ist ebenfalls in Bild 2 zu finden.
Mit Hilfe der bisher erläuterten Abkürzungen läßt sich die verbale Beschreibung der Wertetabelle nach Bild 1 in die Kurzschreibweise

$$Y = \overline{X}_3 * X_2 * X_1 + X_3 * \overline{X}_2 * X_1 + X_3 * X_2 * \overline{X}_1 + X_3 * X_2 * X_1 \quad (2)$$

überführen, welche auch als *boolesche Funktion* oder als *Logikfunktion* bezeichnet wird. Glg.(2) ist sogar eine ganz spezielle Ausprägung einer booleschen Funktion, in welcher *alle* Terme der Wertetabelle, bei denen Y=1 wird, disjunktiv miteinander verknüpft sind. In diesem Fall spricht man von der sog. *disjunktiven Normalform der Logikfunktion*. Die einzelnen Produktterme heißen dann *Minterme* oder auch *Vollkonjunktionen*,

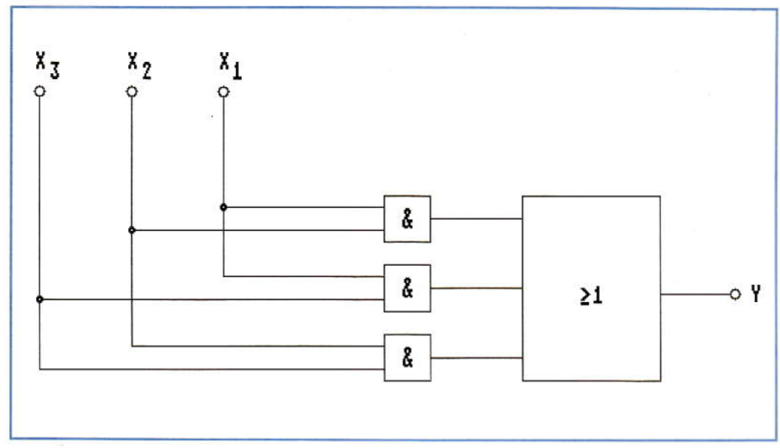



*Bild 4: Blockbilddarstellung von Glg.(11)*

**Die wichtigsten Rechenregeln der booleschen Algebra**

Kommutativgesetz : $X_1 * X_2 = X_2 * X_1$ (4a)
$X_1 + X_2 = X_2 + X_1$ (4b)

Assoziativgesetz : $X_1 * (X_2 * X_3) = (X_1 * X_2) * X_3$ (5a)
$X_1 + (x_2 + X_3) = (X_1 + X_2) + X_3$ (5b)

Distributivgesetz : $X_1 * (X_2 + X_3) = X_1 * X_2 + X_1 * X_3$ (6a)
$X_1 + (X_2 * X_3) = (X_1 + X_2) * (X_1 + X_3)$ (6b)

Tautologie : $X * X = X$ (7a)
$X + X = X$ (7b)

Gesetz für die Negation : $X * \overline{X} = 0$ (8a)
$X + \overline{X} = 1$ (8b)

Doppelte Negation : $\overline{\overline{X}} = X$ (9)

De Morgans Gesetz : $\overline{X_1 * X_2} = \overline{X}_1 + \overline{X}_2$ (10a)
$\overline{X_1 + X_2} = \overline{X}_1 * \overline{X}_2$ (10b)

*Tabelle 1*

weil jeweils *alle* Variablen in jedem der Produktterme vorkommen. Zusammenfassend lauten die Regeln zur Erzeugung der disjunktiven Normalform:

1) Falls noch nicht geschehen, ordnet man die zugrunde liegende Wertetabelle so, daß die Wertekombinationen der Eingangsvariablen $X_1$ bis $X_k$ nach Art von Glg.(1) angeordnet sind, daß also gilt

$$\text{Zeilenzahl} = X_k * 2^{k-1} + X^{k-1} * 2^{k-2} + \ldots + X_2 * 2^1 + X_1 * 2^0. \quad (3)$$

2) Man sucht nun in der Wertetabelle alle Zeilen auf, in denen die Ausgangsvariable Y den Wert 1 besitzt (die zugehörigen Zeilennummern werden als *einschlägige Indizes* bezeichnet).

3) Von jeder dieser Zeilen bildet man die Konjunktion (Und-Verknüpfung) aller Eingangsvariablen. Und zwar setzt man $X_i$ ein, wenn bei der betreffenden Variablen eine 1 steht, andernfalls $\overline{X}_i$. Auf diese Weise erhält man gerade soviele Produktterme wie Zeilen mit Y=1.

4) Die gesuchte Funktion für Y erhält man schließlich, indem man die Disjunktion (Oder-Verknüpfung) aller gefundenen Produktterme bildet.

Wie aus Bild 2 bereits zu erahnen, läßt sich eine boolesche Funktion auch grafisch darstellen, wodurch ein direkter Bezug zur Umsetzung in eine elektronische Schaltung hergestellt wird [1]. Als Beispiel ist in Bild 3 das zu Glg.(2) gehörige Blockschaltbild zu sehen.

Wir kommen nun zur zweiten Hauptaufgabe einer Algebra, der *Angabe von Rechenregeln*. In der arithmetischen Algebra dienen Rechenregeln z.B. zum Lösen von Gleichungen. In der booleschen Algebra versucht man, mit Hilfe der Regeln vorgegebene Logikfunktionen zu vereinfachen. In Tabelle 1 sind die wichtigsten Regeln zusammengestellt [2]. Viele dieser Gesetze sind rein formal bereits aus der arithmetischen Algebra bekannt. Andere wiederum, wie z.B. Glg.(6b), gelten nicht für Zahlen. Wir werden nun versuchen, mit Hilfe der vorgestellten Rechenregeln Glg.(2) zu vereinfachen.

Zunächst fügen wir den letzten Term aus Glg.(2) nochmals disjunktiv hinzu, was uns wegen des Tautologiegesetzes (7b) erlaubt ist. Es entsteht

$$Y = \overline{X}_3 * X_2 * X_1 + X_3 * \overline{X}_2 * X_1 + X_3 * X_2 * \overline{X}_1 + X_3 * X_2 * X_1 + X_3 * X_2 * X_1$$

Nun fassen wir die ersten beiden und den letzten Produktterm zusammen und ziehen vermöge des Distributivgesetzes (6a) die Variable $X_1$ heraus. Damit erhält man

$$Y = X_1 * (\overline{X}_3 * X_2 + X_3 * \overline{X}_2 + X_3 * X_2) + X_3 * X_2 * \overline{X}_1 + X_3 * X_2 * X_1$$

Mit demselben Gesetz können wir $X_3 * X_2$ aus den letzten beiden Termen ausklammern, wodurch sich

$$Y = X_1 * (X_3 * X_2 + X_3 * \overline{X}_2 + X_3 * X_2) + X_3 * X_2 * (\overline{X}_1 + X_1)$$

ergibt, was wegen Glg.(8b) zu

$$Y = X_1 * (\overline{X}_3 * X_2 + X_3 * \overline{X}_2 + X_3 * X_2) + X_3 * X_2$$

vereinfacht werden kann. Erweitern wir den Ausdruck aufgrund des Tautologiegesetzes nochmals auf

$$Y = X_1 * (\overline{X}_3 * X_2 + X_3 * \overline{X}_2 + X_3 * X_2 + X_3 * X_2) + X_3 * X_2$$

können wir die Umformung

$$Y = X_1 * (X_2 * (\overline{X}_3 + X_3) + X_3 * (\overline{X}_2 + X_2)) + X_3 * X_2$$

vornehmen, was wegen Glg.(8b) letztlich auf

$$Y = X_1 * (X_2 + X_3) + X_3 * X_2 = X_1 * X_2 + X_1 * X_3 + X_3 * X_2 \quad (11)$$

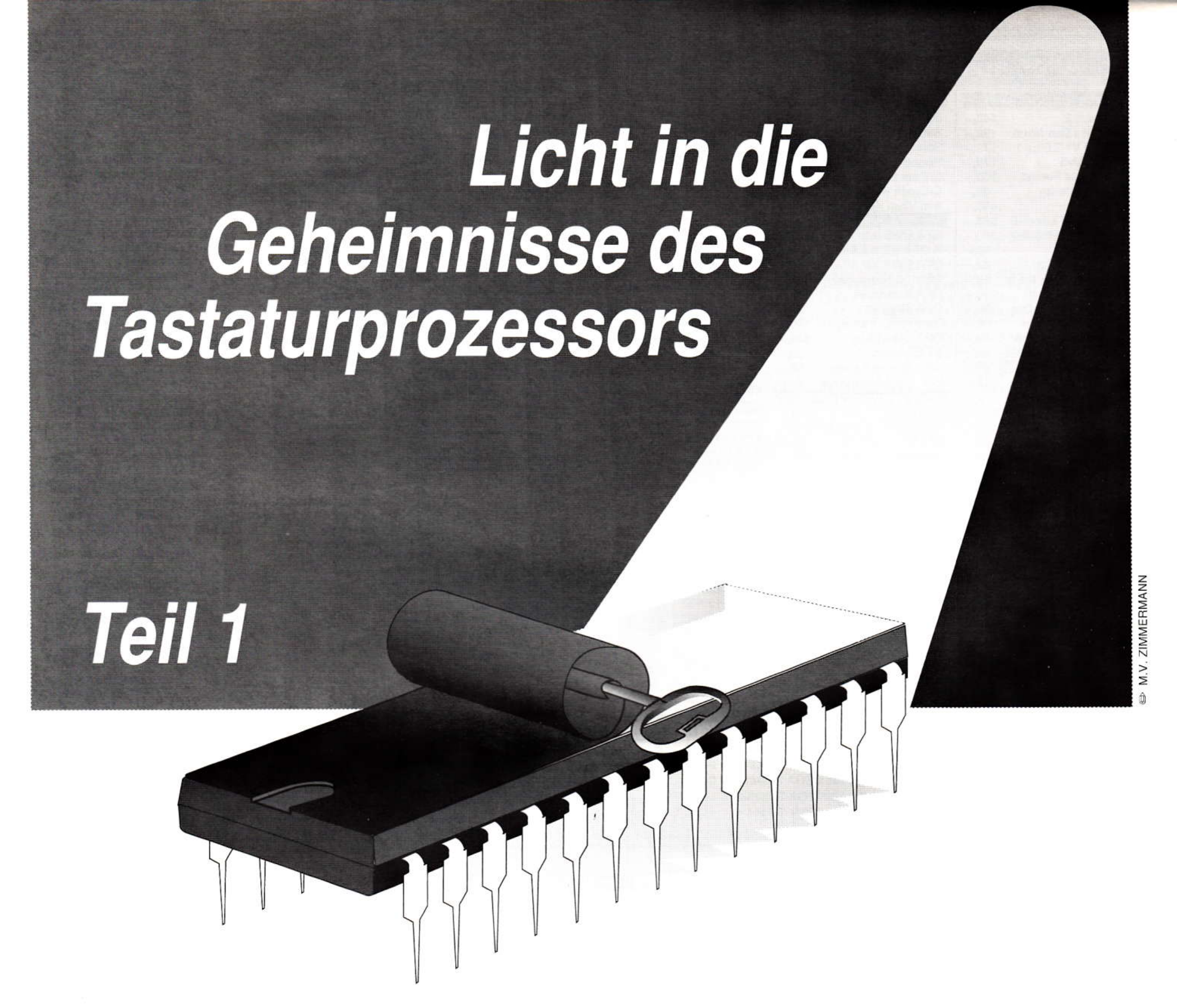
führt. Wir sehen, daß Glg.(11) gegenüber Glg.(2) deutlich vereinfacht ist, was sich auch im Blockschaltbild (siehe Bild 4) durch einen reduzierten Schaltungsaufwand bemerkbar macht.

Andererseits wurde vermutlich deutlich, daß der Einsatz der richtigen Rechenregeln an der richtigen Stelle ein gehöriges Maß an Intuition und Erfahrung erfordern kann. Im nächsten Teil dieses Artikels werden wir das Vereinfachungsverfahren von Quine-McCluskey kennenlernen, welches durch fortwährende Anwendung nur einer einzigen Rechenregel stets imstande ist, den Aufwand einer gegebenen Logikfunktion zu minimieren. Da bei ist keinerlei Intuition notwendig, so daß dieses Verfahren auch auf einem Rechner implementiert werden kann. Das PASCAL-Listing für ein entsprechendes Minimierungsprogramm nach Quine-McCluskey, welches bis zu 19 Eingangsvariablen bearbeiten kann, wird im dritten und letzten Teil zu finden sein.

***Literatur:***

*[1] Werner, T., "Programmierte Logik", Teil 1, ST-Computer, Oktober 1989, pp. 164-167.*

*[2] Tietze, U. und Schenk, Ch., Halbleiter-Schaltungstechnik, Springer-Verlag, 1980.*




# Licht in die Geheimnisse des Tastaturprozessors

## Teil 1

**Die meisten Computer-User wissen im allgemeinen sehr wenig über das Innenleben eines Tastaturprozessors. Dieser dreiteilige Artikel soll deshalb die Neugier derer stillen, die schon immer mehr über das unbekannte Gebiet der vielen Tasten wissen wollten, und den knappen Informationsstand der restlichen Leser auf Vordermann bringen, die lieber die vielen Anwenderprogramme auf dem Markt des ATARI ST zu Rate ziehen, als sich selber mit der Programmierung zu beschäftigen.**

### Assembler-Kenntnisse vorausgesetzt

Zunächst möchte ich aber den vielen Neugierigen erst einmal einen kleinen Dämpfer verpassen, denn so beliebt, wie es die Hochsprachen sind, die es bekanntlich auf fast keinem Rechner so vielzählig wie für unseren ATARI ST gibt, so unbeliebt ist (zum Glück aber nicht immer) der Spaghetti-Code, den der Assembler als stete Nahrung verlangt. Als sich die Digital Research Inc. daran machte, das Betriebssystem des ATARI-Tastaturprozessors in Auftrag zu geben, wußte man von vornherein, daß das nur in Assembler zu bewerkstelligen war. Außerdem ist auf dem Gebiet der Tastaturprozessoren ja die Geschwindigkeit Trumpf - und im Grunde genommen sollte sich kein User darüber sorgen, wie denn das so funktioniert.

Auf den Nenner gebracht, bedeutet das für Sie, liebe Leser, einen harten Brocken an 6301-Assembler, den ich Ihnen aber leider nicht ohne Vorkenntnisse in 68000-Assembler (o.ä.) servieren kann. Ganz interessant wäre bestimmt auch ein wenig Hintergrundwissen, welches sich jeder mit Hilfe von Fachlektüre, wie z.B. dem ATARI ST Profibuch [1], autodidaktisch aneignen kann.

### Was Sie erwartet

Im ersten Teil werde ich also die Register und Adressierungsarten sowie die ersten 82 Befehle (Mnemonics) des HD 6301 V1 (im folgenden nur noch **I**ntelligent **K**ey-**B**oar**D** = IKBD genannt) in der Theorie vorstellen. Der zweite Teil soll dann die auf einen Schlag doch recht unverdauliche Mnemonics-Liste vervollständigen. Mit dem dritten Teil soll dann anhand von Beispielen aus dem disassemblierten IKBD-ROM das Wissen praktisch gefestigt werden, auch die serielle Datenübertragung vom IKBD zum MC68000 wird nicht zu kurz kommen, genausowenig wie die dafür notwendige Erklärung zu den 18 im IKBD enthaltenen Hardware-Registern.

Um aber zum Experimentieren auch ein Werkzeug zur Verfügung zu haben, veröffentliche ich parallel zu diesem Artikel auch ein in den ersten beiden Teilen abgedrucktes Listing des in Frbe und Monochrom arbeitenden Echtzeit-Disassemblers. Dieser unterstützt den kompletten Befehlssatz nach dem HITACHI-Users Manual für den gesamten Speicher des IKBD. Das Listing müssen Sie aber

komplett abgetippt haben, um mit dem Disassembler arbeiten zu können. Deshalb erkläre ich in den ersten beiden Teilen auch nur die theoretischen Zusammenhänge.

## Ein wichtiges Dankeschön

Mein größter Dank gilt von vornherein der Firma HITACHI, ohne deren USER'S MANUAL [2] meine Exkurse ins Reich des IKBD schon im Ansatz gescheitert wären: Freundliche Grüße deshalb vor allem an Herrn Müller vom Technischen Marketing.

Genausowenig vernachlässigen möchte ich aber die Leser meines im April 1989 erschienenen Artikels zur resetfesten Uhr im IKBD [3], die durch ihr Interesse die Wichtigkeit einer Aufklärung kundtaten. Da ich aus Platzgründen kein Listing des disassemblierten und von mir kommentierten IKBD-ROMs veröffentlichen kann, biete ich weiterhin jedem Interessenten an, gegen Einsenden einer formatierten Leerdiskette und Bezahlung des Rückportos (bitte ca. DM 5.- wg. Unkosten) das als ASCII-Datei erstellte Listing bei mir zu erwerben. Die Adresse findet jeder Interessierte im Kopf des Assemblerlistings zum IKBD-Echtzeit-Disassembler.

Gerne erinnere ich mich auch an einen Bekannten, der mich ernsthaft auf einen Artikel in einer Fachzeitschrift hingewiesen hat, der die Programmierung des IKBD als schnellen Co-Prozessor zum 68000 ermöglicht. (April, April). Erklärung (um weiteren Gerüchten Einhalt zu gebieten):

Der IKBD besitzt genau 128 Bytes RAM, das aber für interne Variablen und den Stack belegt wird. Wenn hier jemand einen freien und vor allem zusammenhängenden Platz findet, der nicht bei irgendeiner Übertragung überschrieben wird oder beim Überschreiben gar zum Absturz des IKBD führt, besitzt er ein anderes ROM als das serienmäßig im ATARI ST eingesetzte.

## Jetzt geht's los

Nach dieser Talk-Runde möchte ich mit der Beschreibung einiger interessanter Daten zum IKBD beginnen: Der im ATARI ST mit 4 MHz getaktete HD6301V1 wird als eigenständiger, in CMOS-Technik konstruierter Single-Chip-Prozessor eingesetzt. Intern wird

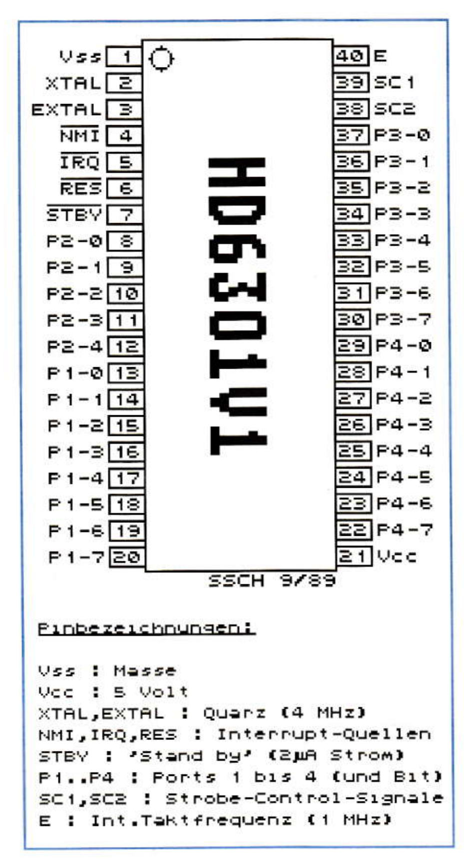



*Abb.1: Pinbelegung des IKBD*

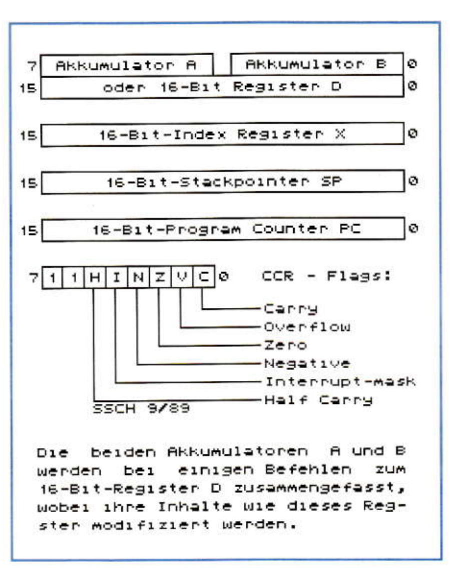



*Abb.2: Die Register im IKBD*

die Taktfrequenz des Oszillators noch einmal geviertelt, so daß effektiv eine Frequenz von 1 MHz zur Verfügung steht. Diese Taktfrequenz kann auch an Pin 40 ("E", s.Abb.1) des IKBD abgegriffen werden. Der Prozessor besitzt 4 kByte ROM, 128 Bytes RAM und 18 zum Teil 16-Bit breite Hardware-Register, die direkt im Speicher adressierbar sind. Sein Betriebssystem wird schon bei der Herstellung fest auf dem Chip "verdrahtet", es ist deshalb nachträglich nicht mehr veränderbar; mit ein Grund, weshalb in allen bis jetzt verkauften ATARI ST-Rechnern immer das gleiche IKBD-ROM eingesetzt wurde.

Bei dem im IKBD verwendeten Modus 7 werden die 29 (!) Port-Leitungen (s.Abb.1) für die Außenwelt zur Verfügung gestellt, in den anderen Modi 0-6 gelten sie neben ein paar übriggebliebenen I/O-Ports dann als Adreß- und Datenbus für eine externe Speichererweiterung. Wie hier leicht zu sehen ist, läßt sich der IKBD im Modus 7 also nicht weiter ausbauen, da seine Ports eben (fast) alle für sein programmiertes Multiplexsystem zur Abfrage der 95 Tasten eingesetzt werden, ebenso noch zusätzlich für die Maus und den Joystick.

Außerdem bietet der IKBD die im ATARI ST lebensnotwendige vollduplex und asynchron arbeitende serielle Schnittstelle, über die die Daten mit 7812.5 Bit/sec zum ST übertragen werden. Bei der für Computer üblichen Spannung von 5 Volt zieht der IKBD nur 6 mA, was eine Verlustleistung von lediglich 30 mW zur Folge hat. Er darf sich damit zu Recht zu den sparsamsten Mikroprozessoren zählen, die heute in großen Mengen erhältlich sind. Der HD6301V1 wird noch in den Typen "A" und "B" produziert, die sich durch höhere Taktraten von 6 bzw. 8 MHz auszeichnen. Die übrigens in vielen ATARI ST-Fachbüchern genannte Tatsache, daß der 6301 zum 8-Bit-6800 aufwärts kompatiblen Objektcode besitzt, trifft nicht immer zu, da einige Befehle (v.a AIM, EIM, OIM und TIM, aber auch andere) auf dem 8-Bit-6800 nicht implementiert sind.

Für alle schon ungeduldig wartenden Leser schwenke ich jetzt über zu den internen Registern des IKBD (s.Abb.2). Vorsicht! ALLE Adressen und Immediates sind nach HITACHI-Konventionen in hexadezimaler Schreibweise, es wird kein '$'-Zeichen mit dargestellt.

## Akkumulatoren und Register im IKBD

Der IKBD besitzt zwei 8-Bit-Akkumulatoren A und B, mit denen die meisten Rechenoperationen durchgeführt werden. Beide zusammen ergeben das 16-Bit-Register D, wobei dann A das High- und B das Low-Byte darstellen. Das zusätzliche 16-Bit-Indexregister X läßt wie beim 68000 einen Speicherzugriff mit Index zu [z.B. X=0080; dann erlaubt der IKBD einen Zugriff auf z.B. (05,X), also 0085]. Der obligatorische 16-Bit Stack-

| LO \ HI | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -- | SBA 1 1 | BRA 2 3 | TSX 1 1 | NEGA 1 1 | NEGB 1 1 | NEG 2 6 X | NEG 3 6 M | SUBA 2 2 # | SUBA 2 3 Z | SUBA 2 4 X | SUBA 3 4 M | SUBB 2 2 # | SUBB 2 3 Z | SUBB 2 4 X | SUBB 3 4 M | 0 |
| 1 | NOP 1 1 | CBA 1 1 | BRN 2 3 | INS 1 1 | -- | -- | AIM 3 7 X | AIM 3 6 Z | CMPA 2 2 # | CMPA 2 3 Z | CMPA 2 4 X | CMPA 3 4 M | CMPB 2 2 # | CMPB 2 3 Z | CMPB 2 4 X | CMPB 3 4 M | 1 |
| 2 | -- | -- | BHI 2 3 | PULA 1 3 | -- | -- | OIM 3 7 X | OIM 3 6 Z | SBCA 2 2 # | SBCA 2 3 Z | SBCA 2 4 X | SBCA 3 4 M | SBCB 2 2 # | SBCB 2 3 Z | SBCB 2 4 X | SBCB 3 4 M | 2 |
| 3 | -- | -- | BLS 2 3 | PULB 1 3 | COMA 1 1 | COMB 1 1 | COM 2 6 X | COM 3 6 M | SUBD 3 3 # | SUBD 2 4 Z | SUBD 2 5 X | SUBD 3 5 M | ADDD 3 3 # | ADDD 2 4 Z | ADDD 2 5 X | ADDD 3 5 M | 3 |
| 4 | LSRD 1 1 | -- | BCC 2 3 | DES 1 1 | LSRA 1 1 | LSRB 1 1 | LSR 2 6 X | LSR 3 6 M | ANDA 2 2 # | ANDA 2 3 Z | ANDA 2 4 X | ANDA 3 4 M | ANDB 2 2 # | ANDB 2 3 Z | ANDB 2 4 X | ANDB 3 4 M | 4 |
| 5 | ASLD 1 1 | -- | BCS 2 3 | TXS 1 1 | -- | -- | EIM 3 7 X | EIM 3 6 Z | BITA 2 2 # | BITA 2 3 Z | BITA 2 4 X | BITA 3 4 M | BITB 2 2 # | BITB 2 3 Z | BITB 2 4 X | BITB 3 4 M | 5 |
| 6 | TAP 1 1 | TAB 1 1 | BNE 2 3 | PSHA 1 4 | RORA 1 1 | RORB 1 1 | ROR 2 6 X | ROR 3 6 M | LDAA 2 2 # | LDAA 2 3 Z | LDAA 2 4 X | LDAA 3 4 M | LDAB 2 2 # | LDAB 2 3 Z | LDAB 2 4 X | LDAB 3 4 M | 6 |
| 7 | TPA 1 1 | TBA 1 1 | BEQ 2 3 | PSHB 1 4 | ASRA 1 1 | ASRB 1 1 | ASR 2 6 X | ASR 3 6 M | -- | STAA 2 3 Z | STAA 2 4 X | STAA 3 4 M | -- | STAB 2 3 Z | STAB 2 4 X | STAB 3 4 M | 7 |
| 8 | INX 1 1 | XGDX 1 2 | BVC 2 3 | PULX 1 4 | ASLA 1 1 | ASLB 1 1 | ASL 2 6 X | ASL 3 6 M | EORA 2 2 # | EORA 2 3 Z | EORA 2 4 X | EORA 3 4 M | EORB 2 2 # | EORB 2 3 Z | EORB 2 4 X | EORB 3 4 M | 8 |
| 9 | DEX 1 1 | DAA 1 2 | BVS 2 3 | RTS 1 5 | ROLA 1 1 | ROLB 1 1 | ROL 2 6 X | ROL 3 6 M | ADCA 2 2 # | ADCA 2 3 Z | ADCA 2 4 X | ADCA 3 4 M | ADCB 2 2 # | ADCB 2 3 Z | ADCB 2 4 X | ADCB 3 4 M | 9 |
| A | CLV 1 1 | SLP 1 4 | BPL 2 3 | ABX 1 1 | DECA 1 1 | DECB 1 1 | DEC 2 6 X | DEC 3 6 M | ORAA 2 2 # | ORAA 2 3 Z | ORAA 2 4 X | ORAA 3 4 M | ORAB 2 2 # | ORAB 2 3 Z | ORAB 2 4 X | ORAB 3 4 M | A |
| B | SEV 1 1 | ABA 1 1 | BMI 2 3 | RTI 1 10 | -- | -- | TIM 3 5 X | TIM 3 4 Z | ADDA 2 2 # | ADDA 2 3 Z | ADDA 2 4 X | ADDA 3 4 M | ADDB 2 2 # | ADDB 2 3 Z | ADDB 2 4 X | ADDB 3 4 M | B |
| C | CLC 1 1 | -- | BGE 2 3 | PSHX 1 5 | INCA 1 1 | INCB 1 1 | INC 2 6 X | INC 3 6 M | CPX 3 3 # | CPX 2 4 Z | CPX 2 5 X | CPX 3 5 M | LDD 3 3 # | LDD 2 4 Z | LDD 2 5 X | LDD 3 5 M | C |
| D | SEC 1 1 | -- | BLT 2 3 | MUL 1 7 | TSTA 1 1 | TSTB 1 1 | TST 2 4 X | TST 3 4 M | BSR 2 5 | JSR 2 5 Z | JSR 2 5 X | JSR 3 6 M | -- | STD 2 4 Z | STD 2 5 X | STD 3 5 M | D |
| E | CLI 1 1 | -- | BGT 2 3 | WAI 1 9 | -- | -- | JMP 2 3 X | JMP 3 3 M | LDS 3 3 # | LDS 2 4 Z | LDS 2 5 X | LDS 3 5 M | LDX 3 3 # | LDX 2 4 Z | LDX 2 5 X | LDX 3 5 M | E |
| F | SEI 1 1 | -- | BLE 2 3 | SWI 1 12 | CLRA 1 1 | CLRB 1 1 | CLR 2 5 X | CLR 3 5 M | -- | STS 2 4 Z | STS 2 5 X | STS 3 5 M | -- | STX 2 4 Z | STX 2 5 X | STX 3 5 M | F |

SSCH 11/89

LEGENDE: Zahlen unter den Mnemonics : Links = Anzahl der Bytes pro Befehl
Mitte = Anzahl der Zyklen des Befehls (1 Zyklus dauert 1µs)
Die Symbole '#' 'Z' 'X' und 'M' beschreiben die Adressierungsart des Befehls wie folgt:

'#' : es folgt ein Immediate-Wert
'Z' : Speicherzugriff auf die Zeropage
'X' : Index-Wert zum 16-Bit-Register X
'M' : direkter 16-Bit-Speicherzugriff

HD6301V1

*Tabelle 1: Die 230 Mnemonics des Tastaturprozessors*

pointer SP und der 16-Bit Programcounter PC runden mit dem 8-Bit-Condition-Code-Register CCR die Palette der Prozessorregister ab.

## Die Flags des CCR

*C* "Carry" (Übertrag bei Schiebe- und Rechenoperationen)

*V* "Overflow" (Übertrag bei vorzeichenbehafteten Rechenoperationen)

*Z* "Zero" (gesetzt, wenn das Rechenergebnis Null ist)

*N* "Negative" (auf 1, wenn das "höchstwertige Bit" MSB gesetzt, das Rechenergebnis also vorzeichenbehaftet und logisch negativ ist)

*I* "Interrupt Mask" (die Interrupts IRQ1 (extern), IRQ2 (interner Timer und serielle Schnittstelle, mehr dazu im Teil 3) und der Software-Interrupt 'SWI' werden alle mit diesem Bit maskiert oder freigegeben)

*H* "Half Carry" (ein Carry nur für das Low-Nibble; es wird gesetzt, wenn bei einer Rechenoperation ein übertrag vom Low- zum Highnibble stattfand).

## Die Adressierungsarten

Der IKBD erlaubt insgesamt sieben Adressierungsarten, das ist einiges weniger als beim 68000 - macht uns die Arbeit aber nicht leichter:

*1. Accumulator Addressing:* Operationen nur in den Akkumulatoren A oder B (z.B. CLRA - lösche Akkumulator A).

*2. Implied Addressing:* Operationen mit den Registern A,B,D,X,SP und CCR untereinander (z.B. ABX - addiere den 8-Bit-Wert von Akkumulator B zum 16-Bit-Register X).

*3. Immediate Addressing:* Operationen, die mit einem direkt angegebenen Wert (Immediate, immer durch '#' gekennzeichnet) arbeiten (z.B. ANDA #7F - verknüpfe den Inhalt von Akkumulator A mit #7F durch logisches UND).

*4. Direct Addressing* und

*5. Extended Addressing:* Operationen mit direktem Zugriff auf Inhalte des Speichers. Dabei bedeutet 'direct' einen 8-Bit-Zugriff auf die untersten 256 Bytes von 0000 bis 00FF (auch bekannt als Zeropage und deshalb ab jetzt so genannt) und 'extended' einen 16-Bit-Zugriff auf den gesamten Speicher (z.B. 'direct': LDAA C2, lade Inhalt von Speicherzelle 00C2 nach Akkumulator A - 'extended': LDAB F206 - lade Inhalt von Speicherzelle F206 nach Akkumulator B).

*6. Indexed Addressing:* Gegenüber 4. und 5. besteht hier die Möglichkeit, mit Hilfe des Registers X einen relativen Zugriff auf den Speicher auszuführen (z.B. LDAB 05,X — angenommen X=0080, dann würde hier der Inhalt von Speicherzelle 0085, also von 05 Bytes über X, nach B geladen werden).

*7. Relative (and Absolute)* Addressing: Diese Adressierungsart gilt für alle relativen und absoluten Sprünge im Speicher (z.B. BRA 1E - springe an die Stelle, die 1E relativ zur jetzigen, vom PC angezeigten liegt oder JSR F8D4 - springe direkt nach F8D4 und führe dort ein Unterprogramm aus).

## Die Mnemonics - darauf hat jeder gewartet

In Tab.1 sehen Sie alle 230 Mnemonics auf einen Blick, schön geordnet. In [2] sind die Befehle (allerdings in Englisch) wesentlich ausführlicher erläutert, ich empfehle hier bei weitergehendem Inter-

esse den Erwerb des MANUALs von HITACHI. Anmerken möchte ich nochmals, daß sich die Zeropage auf die untersten 256 Bytes im Speicher bezieht; hier kann der IKBD also mit nur 8-Bit-Adressenangabe die eigentlich 16-Bit breiten Adressen von 0000 bis 00FF speichersparend und schnell ansprechen.

## In alphabetischer Reihenfolge

*ABA Add Accumulator B to Accumulator A*
Die 8-Bit-Summe aus den Akkumulatoren A und B wird in Akkumulator A gespeichert.

*ABX Add Accumulator B to Index Register*
Die Summe vom 16-Bit-Register X und dem 8-Bit Akkumulator B wird in X gespeichert.

*ADCA Add Accumulator A with Carry*
Hier wird ein Immediate, der Inhalt einer Zeropage- oder 16-Bit-Speicherzelle oder der Inhalt der relativ zum 16-Bit-Register X angegebenen Speicherzelle zum Akkumulator A 8-Bit-addiert. Grundsätzlich wird das Carry-Bit immer mitaddiert.

*ADCB Add Accumulator B with Carry*
s.ADCA, nur mit Akkumulator B

*ADDA Add Accumulator A without Carry*
Wie bei ADCA wird hier aber grundsätzlich ohne das Carry-Bit addiert.

*ADDB Add Accumulator B without Carry*
s.ADDA, nur mit Akkumulator B

*ADDD Double Add without Carry*
Hier werden die Akkumulatoren A und B zum Register D intern zusammengefaßt (s.Abb.1) und entweder ein angegebener 16-Bit-Immediate-Wert oder der Inhalt einer Zeropage- oder 16-Bit-Speicherzelle oder der Inhalt der relativ zum 16-Bit-Register X angegebenen Speicherzelle zu D 16-Bit-addiert.

*AIM And IMmediate*
Der Inhalt einer Speicherzelle der Zeropage oder der Inhalt der relativ zu X angegebenen Speicherzelle wird mit dem als Immediate angegebenen Byte logisch verUNDet.

*ANDA Logical AND with A*
Der Inhalt von A wird mit dem Inhalt einer Zeropage- oder einer 16-Bit-Speicherzelle oder einem angegebenen Immediate-Byte oder mit dem einer relativ zu X angegebenen Speicherzelle logisch verUNDet.

*ANDB Logical AND with B*
s.ANDA, nur mit Akkumulator B

*ASLA Arithmetic Shift Left A*
Der Akkumulator A wird um 1 Bit nach links geschoben. Das herausfallende Bit kommt ins Carry-Flag. Rechts wird eine Null eingeschoben.

*ASLB Arithmetic Shift Left B*
s.ASLA, nur mit Akkumulator B

*ASL Arithmetic Shift Left*
Der Inhalt (Byte) einer 16-Bit-Speicherzelle oder einer relativ zu X angegebenen Speicherzelle wird um ein Bit nach links verschoben. Das herausfallende Bit kommt ins Carry-Flag. Rechts wird eine Null eingeschoben.

*ASLD Arithmetic Double Shift Left D* (A und B)
Der Inhalt der zum 16-Bit-Register D zusammengefaßten Akkumulatoren A und B wird um 1 Bit nach links verschoben. Das herausfallende Bit kommt ins Carry-Flag. Rechts wird eine Null eingeschoben.

*ASRA Arithmetic Shift Right A*
s.ASLA, nur nach rechts. Das MSB (Bit 7) wird beibehalten.

*ASRB Arithmetic Shift Right B*
s.ASLA, nur Akkumulator B nach rechts. Das MSB wird beibehalten.

*ASR Arithmetic Shift Right*
s.ASL, nur nach rechts. Das MSB wird beibehalten.

Die bedingten Sprünge sind identisch mit denen des 68000. Deshalb habe ich hier nur die zusätzlich vorhandenen Befehle erläutert:

*BCC Branch if Carry Clear*
*BCS Branch if Carry Set*
*BEQ Branch if EQual*
*BGE Branch if Greater than or Equal to zero*
*BGT Branch if Greater Than to zero*
*BHI Branch if HIgher*
*BLE Branch if Less than or Equal to zero*
*BLS Branch if Lower or Same*
*BLT Branch if Less Than zero*
*BMI Branch if MInus*
*BNE Branch if Not Equal*
*BPL Branch if PLus*
*BRA BRanch Always*
*BRN BRanch Never (identisch mit NOP)*
*BSR Branch SubRoutine*

*BVC Branch if oVerflow Clear*
Springt, wenn das Overflow-Flag V gelöscht ist.

*BVS Branch if oVerflow Set*
Springt, wenn das Overflow-Flag V gesetzt ist.

*BITA BIt Test A*
Führt ein logisches UND aus, ohne die beiden Argumente zu verändern. Die Flags N und Z werden wie beim 68000 verändert. Getestet werden können Bits eines direkt angegebenen Immediates, der Inhalt einer Zeropage- oder einer 16-Bit-Speicherzelle oder der Inhalt einer relativ zu X angegebenen Speicherzelle.

*BITB BIt Test B*
s.BITA, nur mit Akkumulator B.

*CBA Compare Accumulators*
Die beiden Akkumulatoren A und B werden miteinander verglichen. Die Flags C,N,V,Z werden wie bei einer virtuellen Subtraktion von (A minus B) gesetzt.

*CLC CLear Carry*
Das Carry-Flag C wird auf 0 gesetzt.

*CLI CLear Interrupt mask*
Das Interrupt-Flag I wird auf 0 gesetzt. Alle Interrupt-Quellen werden jetzt an den IKBD durchgelassen.

*CLRA CLeaR A*
Der Akkumulator A wird gelöscht.

*CLRB CLeaR B*
Der Akkumulator B wird gelöscht.

*CLR CLeaR*
Hier wird entweder der Inhalt einer 16-Bit-Speicherzelle oder der Inhalt einer relativ zu X angegebenen Speicherzelle auf 8-Bit-Breite gelöscht.

*CLV CLear two's complement oVerflow bit*
Setzt das Overflow-Flag V auf 0.

*CMPA CoMPare with A*
Vergleicht den Inhalt von Akkumulator A entweder mit einem angegebenen Immediate-Wert oder mit dem Inhalt einer Zeropage- oder einer 16-Bit-Speicherzelle oder mit dem Inhalt einer relativ zu X angegebenen Speicherzelle. Die Flags C,N,V,Z werden so gesetzt, als ob der angegebene Wert vom Akkumulator A abgezogen würde.

*CMPB CoMPare with B*
s.CMPA, nur mit Akkumulator B.

## Das war's für heute

Mehr kann und will ich Ihnen nicht auf einmal zumuten. Im nächsten Teil werde ich also die restlichen 148 Befehle ansprechen, um Sie dann gleich nach dem Abtip-

## PLOTMATH für ATARI ST DM 498.-

mathematische + grafische MESSDATENVERARBEITUNG

J&M Analytische Meß- und Regeltechnik GmbH · 7080 Aalen
Gartenstraße 133 · Tel. 07361/45 111 · Fax 07361/45 187

Daten-/Plotausgabe : 9/24-Nadeldrucker, HPGL-Plotter, ASCII
Dateneingabe : ASCII, Maus, Tastatur
optional : Schnittstelle zum c't - IBM-Bus-Interface

- Autoskalierung, Zoom, vielfältige Beschriftung und Achsenformate
- 3D-Grafik (Hidden line)
- Rauschgenerator
- Funktionsgenerator/-parser
- Kommandospracheninterpreter

- FFT-Routinen, digitale Filter
- lineare, nichtlineare Regression
- Differentiation / Integration
- Savitzky-Golay-Algorithmus
- schnellste Spline-Interpolation
- u.v.a.m.

*Einmalig :* Optimale Spline-Glättung !
Ausführliches Info und Demo-Diskette kostenlos

## ST-Floppy-Stationen:

■ anschlußfertig ■ doppelseitig ■ garantiert kompatibel ■ ■ mit formschönem, atari-grauem Metallgehäuse ■ mit dem bewährten NEC FD1036A ■ mit eingebautem Netzteil, und dem einzigartigen automatischen Netzschalter ■ ■ mit Netzkontrollleuchte ■■■ 12 Monate Garantie ■■■

| | | |
|---|---|---|
| ESN: | 3,5"- Einzelstation, 42 ∗ 108 ∗ 230 mm, als B-Laufwerk z.B. am 1040 ST | 249,-- DM |
| ESN/A: | dto mit Ausgang für Laufwerk B, als A-Laufwerk am 520 ST | 268,-- DM |
| DSN: | 3,5"- Doppelstation, 75 ∗ 108 ∗ 230mm | 398,-- DM |

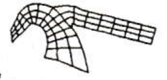
**Dipl.Ing. Gerhard Trumpp**
**Mitterlängstrasse 7**
**8039 Puchheim - Ort**
**Tel. 089 / 80 68 23**

## Finite Elemente



für Personal Computer

**- - Z88 - -**



Das FE-Programm Z88 löst statische Tragwerksaufgaben für den ebenen, den axialsymmetrischen und den räumlichen Spannungszustand unter ATARI GEMTOS, MS-DOS und OS/2.
Zur Beschreibung einer 2-dimensionalen, axialsymetrischen oder 3-dimensionalen Struktur stehen 12 Elementtypen zur Verfügung.
Das FE- System Z88 ist seit über zwei Jahren in der Industrie im Einsatz. Codiert in FORTRAN 77. Weitergehende Informationen, Infoblatt, Systemvoraussetzungen und Bestellungen bei:

Lizenz-Preise zzgl. Versand :
Atari ST, mind. 1 MByte : 198,00 DM
Atari Mega ST4, IBM PC/XT/AT, MS-DOS : 498,00 DM
IBM AT, OS/2, 386er ( 386 zzgl. OS386 ) : 598,00 DM
Handbuch einzeln : 48,00 DM
Das Handbuch wird beim Kauf voll angerechnet.
Bei Bestellungen bitte Diskettenformat mit angeben!.

**HPS GmbH**
Karlsbader Str. 10
6100 Darmstadt
Telefon 06151 / 316132

## *Textverarbeitung am Atari ST*

Wenn Sie auf optimale Druckqualität Wert legen, mußten Sie bisher meist viel Geld und Arbeit investieren. Script könnte für Sie eine interessante Alternative darstellen. Wir

SCRIPT

haben uns mehrere Wochen intensiv mit den Möglichkeiten des Programms befaßt und seine Grenzen ausgelotet. Unter dem Stichwort "Script Erfahrungsbericht" können Sie kostenlos unsere Unterlagen anfordern.

ALTEX Textsysteme Tel: 0871/78496
Dipl. Ing. Georg Altmann, Dekan Simbürger Str. 13, 8300 Ergolding

# Entscheiden Sie selbst!

**Bitte ankreuzen:**

Wollen Sie anspruchsvolle ST Games? Ja ☐ Nein ☐
Wollen Sie preisgünstige ST Games? Ja ☐ Nein ☐

Ergebnis: 2 x Ja - Prüfen Sie unser Angebot !

### Rollenspiele

| | |
|---|---|
| Bloodwych | 69,90 |
| Kult | 67,90 |
| Sleeping Gods Lie | 65,90 |

### Simulation

| | |
|---|---|
| Pirates | 65,90 |
| Stuntcar | 65,90 |
| F-16 Combat Pilot | 65,90 |

### Sport

| | |
|---|---|
| Microprose Soccer | 63,90 |
| Buffalo Bill's Rodeo | 65,90 |
| Passing Shot (Tennis) | 55,90 |
| TV Sports Football | 65,90 |

### Adventure

| | |
|---|---|
| Manic Mansion (dtsch.) | 72,90 |
| Indiana Jones (dtsch.) -The last Crusade | 72,90 |

### Action

| | |
|---|---|
| Running Man (dt.) | 63,90 |
| Xenon II Megablast | 69,90 |
| Blood Money | 65,90 |
| Targhan | 65,90 |

### Aktuell

| | |
|---|---|
| Chessplayer 2150 | 63,90 |
| Ghostbusters 2 | 69,90 |
| Borodino | 77,90 |
| Fighting Soccer | 59,90 |
| Turbo Outrun | 59,90 |

### Happy Games

| | |
|---|---|
| Rick Dangerous | 65,90 |
| Paperboy | 53,90 |

### Strategie

| | |
|---|---|
| Balance of Power 1990 | 65,90 |
| Waterloo | 65,90 |

Nicht vergessen - kostenlosen **Gesamtkatalog** anfordern (enthält auch günstige PD- Software) !

### SIERRA total

| | |
|---|---|
| Goldrush | 65,90 |
| King's Quest 1/2/3 | 84,90 |
| King's Quest 4 | 72,90 |
| Larry 1 | 55,90 |
| Larry 2 | 72,90 |
| Manhunter 1 | 72,90 |
| Manhunter 2 | 79,90 |
| Police Quest 1 | 55,90 |
| Police Quest 2 | 65,90 |
| Space Quest 1 | 65,90 |
| Space Quest 2 | 55,90 |
| Space Quest 3 | 72,90 |

Bestellung
07252/3058



Komplettlösungen zu diesen Sierra-Spielen: **DM 12,-** je Lösung. Alle 14 komplett im Ringbuchordner **DM 79,-**

**Neu!** Manhunter 2 mit Komplettlösung für nur **DM 85,-**

Sofort bestellen bei:
**Werner Rätz,**
**Postfach 1640/ST,**
**7518 Bretten**




POWER
PER
POST

Bei Fragen zu Sierra Adventures, einfach anrufen !

Die Lieferung erfolgt per Nachnahme, zuzügl. DM 6,50 (Ausland DM 10,-) oder per Vorauskasse, zuzügl. DM 4,- (Ausland DM 6,-).

# Richtige Entscheidung!

pen des Disassembler-Listings im Inneren des IKBD herumstochern zu lassen.

Eine kurze Warnung muß ich hier zum Schluß nochmal loswerden: Das Listing wurde in zwei Teile aufgeteilt und ist nicht vor dem kompletten Abtippen lauffähig! Eine kleine Anleitung folgt deshalb erst im zweiten Teil dieser Artikelserie. Ich hoffe, Ihnen ein klein wenig bei der Entdeckungsreise in die Geheimnisse des Tastaturprozessors geholfen zu haben und würde mich freuen, wenn Sie weiter am Ball blieben.

*Sieghard Schäfer*


***Literatur:***

*[1] Jankowski, Rabich, Jeschke : ATARI ST Profibuch, Sybex Verlag Düsseldorf*

*[2] HITACHI USER'S MANUAL (Nr.12-20) zum "HD 6301 V1", HITACHI ELECTRONIC COMPONENTS EUROPE GMBH, München*

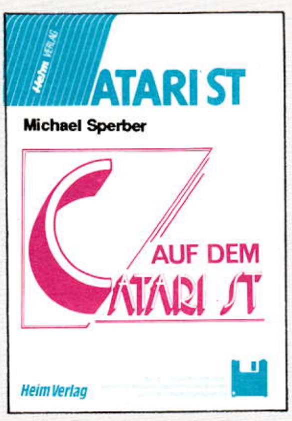
*[3] Artikel "Anwendung des Tastaturprozessors", ST-Computer 4/89, S.141ff*


```
1:   **************************************************
2:   * Dies ist der 1.Teil des Disassembler-Listgs z.*
3:   * dreitlg. Serie "Licht in die Geheimnisse des  *
4:   * Tastaturprozessors".                           *
5:   *               Sieghard Schäfer                 *
6:   *                Jahnstrasse 33                  *
7:   *               7400  Tübingen 5                 *
8:   *          (c) MAXON Computer GmbH 1990          *
9:   *                                                *
10:  *          nur zusammen mit Teil 2 lauffähig     *
11:  * geschrieben m. DevpacST-Assembler 2.0 (HiSoft)*
12:  **************************************************
13:
14:  start    move     #4,-(a7)
15:   trap    #14                 Auflösung holen
16:   addq.l  #4,a7
17:   tst     d0
18:   bne.s   no_low              MID oder HIGH?
19:
20:   lea     no_work(pc),a5      nein, dann Text..
21:   bsr     txt_out             ..ausgeben,..
22:   bsr     tast                ..auf Taste warten..
23:   bra     fin                 ..und ENDE
24:
25:  no_low   lea      rez(pc),a0
26:   move.b  d0,(a0)             Auflösung abspeichern
27:
28:   lea     text(pc),a5         Bildschirm aufbauen
29:   bsr     txt_out
30:
31:   move    #3,-(a7)
32:   trap    #14                 LOGBASE
33:   lea     scrnptr(pc),a0
34:   move.l  d0,(a0)             Bildschirmstartadresse
35:
36:   move    #34,-(a7)
37:   trap    #14                 KBDVBASE
38:   addq.l  #4,a7               Stack korrigieren
39:   move.l  d0,a0               nach a0 kopieren
40:   lea     savea(pc),a1
41:   move.l  a0,(a1)             a0 in savea retten
42:
43:   lea     intrpt(pc),a1       'intrpt'-Routine in a1
44:   lea     save(pc),a3
45:   move.l  12(a0),(a3)         alt.Statusvektor retten
46:   move.l  a1,12(a0)           auf 'intrpt' verbiegen
47:
48:   lea     cr(pc),a0
49:   move    #$a0d,(a0)          LF und CR zur Ausgabe..
50:   lea     anfang(pc),a4       ..der ersten 23..
51:   move.b  #22,(a4)            ..Zeilen anhängen
52:
53:  loop     lea      flag(pc),a3
54:   sf      (a3)                flag löschen
55:   pea     send(pc)            Startadresse ist send
56:   pea     $190002             3 Bytes senden
57:   trap    #14                 IKBDWS
58:   addq.l  #8,a7
59:
60:  wait     move.b   (a3),d0
61:   beq.s   wait                noch nichts angekommen
62:
63:   cmp.b   #$20,d0             sind es Memory-Daten?
64:   bne.s   loop                nein, dann nochmal
65:
66:   lea     string(pc),a0       String für eine Zeile
67:
68:   moveq   #32,d0
69:  blanks   move     #'  ',(a0,d0.w) String löschen
70:   subq.b  #2,d0
71:   bpl.s   blanks
72:
73:   moveq   #':',d0
74:   move.b  d0,5(a0)            ':' übergeben
75:   move.b  d0,16(a0)
76:
77:   clr.l   d6
78:   lea     adresse(pc),a1      IKBD-Adresse in Hexa-..
79:   move.b  (a1)+,d7            ..dezimal ausgeben
80:   bsr     convert             Hi-Byte
81:   move.b  (a1),d7             Lo-Byte
82:   bsr     convert
83:
84:   clr.l   d0
85:   lea     daten(pc),a6        Code des Mnemonics..
86:   move.b  (a6),d0             ..errechnen
87:
88:   move.b  d0,d7               auszugebendes Byte d7
89:   moveq   #7,d6               ab String-Stelle d6
90:   bsr     convert             Ausgabe der Code-Nr.
91:
92:   move    d0,d1
93:   move    d0,d2
94:   lsr     #2,d1               durch 4 teilen
95:   lea     len(pc),a1          Längentabelle
96:   move.b  (a1,d1),d1          d1 enthält Code-Länge
97:
98:   and.b   #3,d2               ausmaskieren,..
99:   add.b   d2,d2               ..verdoppeln und..
100:  lsr.b   d2,d1               ..verschieben
101:
102:  and.b   #3,d1               wieder maskieren
103:  lea     anzahl(pc),a4
104:  move.b  d1,(a4)             abspeichern
105:
106: ********************** BRANCHES ****************
107:
108:  cmpi.b  #$1f,d0             > $1f ?
109:  bls.s   no_bra
110:
111:  cmpi.b  #$30,d0             <= $30 ?
112:  bls.s   is_bra
113:
114:  cmpi.b  #$8d,d0             BSR ?
115:  bne.s   no_bra
116:
117: is_bra   move.b   #'>',28(a0)
118:  move.b  1(a6),d1            Rel.Adr.des BRA in d1
119:  ext     d1                  wg.Vorzeichen erweitern
120:  move    adresse(pc),d2      Aktuelle Speicheradr.
121:  add     d1,d2
122:  addq    #2,d2               +Offset zur BRA-Adresse
123:
124:  move.b  d2,d7
125:  moveq   #32,d6
126:  bsr     convert             Low-Byte und..
127:
128:  lsr     #8,d2
129:  move.b  d2,d7
130:  moveq   #30,d6
131:  bsr     convert             ..High-Byte d.Sprung-..
132:  bra     not_aim             ..adresse darstellen
133:
134: ********************** IMMEDIATE ***************
135:
136: no_bra   cmpi.b   #$cd,d0            unbekannt?
137:  beq     no_out
138:
139:  move.b  d0,d1
140:  and.b   #$f0,d1             HI-Nibble ausmaskieren
141:  cmpi.b  #$80,d1             HI-Nibble = 8?
142:  beq.s   file
143:
144:  cmpi.b  #$c0,d1             HI-Nibble = C?
145:  bne.s   no_imm
146:
```

→

```
147:    move.b   d0,d1
148:    moveq    #7,d2
149:    and.b    d2,d1
150:    sub.b    d2,d1                 unbekannter Befehl?
151:    beq      no_out
152:
153:  file       move.b   #'#',23(a0)      '#' f. Immediate
154:    bra      not_aim
155:
156:  **************** DIRECT (Zeropage) ************
157:
158:  no_imm     cmpi.b   #$90,d1        HI-Nibble = 9?
159:    beq.s    zero
160:
161:    cmpi.b   #$d0,d1               HI-Nibble = D?
162:    bne.s    no_zero
163:
164:  zero       move     #'00',24(a0)     00 f. Zeropage..
165:    lea      zerflag(pc),a5
166:    st       (a5)                  ..und zerflag setzen
167:    bra      not_aim
168:
169:  *********************** INDEXED *****************
170:
171:  no_zero    cmpi.b   #$a0,d1        HI-Nibble = A?
172:    beq.s    index
173:
174:    cmpi.b   #$e0,d1               HI-Nibble = E?
175:    beq.s    index
176:
177:    cmpi.b   #$60,d1               HI-Nibble = 6?
178:    bne.s    no_indx
179:
180:  index      move     #',X',26(a0)     Indexsy. setzen
181:
182:  ****************** AIM OIM EIM TIM ************
183:
184:  no_indx    move.b   d0,d1
185:    or.b     #$1f,d1
186:    cmpi.b   #$7f,d1
187:    bne.s    not_aim
188:
189:    move.b   d0,d1
190:    and.b    #$f,d1
191:    cmpi.b   #1,d1                 AIM?
192:    beq.s    aim
193:
194:    cmpi.b   #2,d1                 OIM?
195:    beq.s    aim
196:
197:    cmpi.b   #5,d1                 EIM?
198:    beq.s    aim
199:
200:    cmpi.b   #$b,d1                TIM?
201:    bne.s    not_aim
202:
203:  aim        move.b   #'#',23(a0)      '#' f. Immediate
204:
205:    moveq    #24,d6
206:    move.b   1(a6),d7
207:    bsr      convert               Hex-Code und..
208:
209:    moveq    #10,d6
210:    move.b   1(a6),d7              ..Immediate-Byte..
211:    bsr      convert               ..ausgeben
212:
213:    moveq    #13,d6
214:    move.b   2(a6),d7              3.Byte des Befehls..
215:    bsr      convert               ..ausgeben
216:
217:  *************** Immediate auf Zeropage *********
218:
219:    cmpi.b   #$71,d0               AIM?
220:    beq.s    aim1
221:
222:    cmpi.b   #$72,d0               OIM?
223:    beq.s    aim1
224:
225:    cmpi.b   #$75,d0               EIM?
226:    beq.s    aim1
227:
228:    cmpi.b   #$7b,d0               TIM?
229:    bne.s    no_aim1
230:
231:  aim1       move.l   #',00 ',26(a0)  Zeropage-HIB'00'
232:    moveq    #29,d6
233:    bra.s    add
234:
235:  ************** Immediate mit Offset auf X *******
236:
237:  no_aim1    moveq    #',',d6          ',' und..
238:    move.b   d6,26(a0)
239:    move.b   d6,29(a0)
240:    move.b   #'X',30(a0)           ..'X' ausgeben
241:    moveq    #27,d6
242:    bra.s    add
243:
244:  ************************* ENDE *****************
245:
246:  not_aim    lea      anzahl(pc),a1
247:    move.b   (a1),d5               Bytelänge des Befehls..
248:    subq.b   #1,d5                 ..gleich 1?
249:    beq.s    no_out
250:
251:    moveq    #10,d6                nein, dann restliches..
252:    move.b   1(a6),d7              ..Byte darstellen
253:    bsr      convert
254:
255:    lea      zerflag(pc),a1
256:    tst.b    (a1)                  in der Zeropage?
257:    bmi.s    itszero
258:
259:    moveq    #24,d6                nein,..
260:    bra.s    ok
261:
262:  itszero    moveq    #26,d6          ..2 Stellen weiter..
263:    sf       (a1)                  (zerflag löschen)
264:
265:  ok         move.b   1(a6),d7        ..das Byte nochmal..
266:    bsr      convert               ..darstellen
267:
268:    subq.b   #1,d5                 ein drittes Byte?
269:    beq.s    no_out
270:
271:    moveq    #13,d6                ja, auch dieses..
272:    move.b   2(a6),d7              .. hier und..
273:    bsr      convert
274:
275:    moveq    #26,d6                hier noch einmal..
276:  add        move.b   2(a6),d7        ..darstellen
277:    bsr      convert
278:
279:  ***********************************************
280:
281:  no_out     lea      codes(pc),a1    Stelle des Mnemo
282:    add      d0,d0                 ..berechnen und..
283:    add      d0,d0                 ..in String..
284:    move.l   (a1,d0.w),18(a0)      ..kopieren
285:
286:    lea      string(pc),a5
287:    bsr      txt_out               String-Ausgabe
288:
289:    lea      adresse(pc),a0        Adresse des nächsten..
290:    move.b   anzahl(pc),d0         ..Befehls ermitteln..
291:    ext      d0
292:    add      d0,(a0)               ..und abspeichern
293:
294:    lea      anfang(pc),a0         die ersten 23 Zeilen..
295:    tst.b    (a0)                  ..für Fullpage..
296:    beq.s    weiter
297:
298:    subq.b   #1,(a0)               ..schon angezeigt?
299:    bne      loop                  nein, nächste Zeile
300:
301:    lea      cr(pc),a4             sonst ab jetzt nur..
302:    clr      (a4)                  ..noch jede Zeile..
303:    bra      loop                  ..einzeln darstellen
304:
305:  weiter     bsr      tast
306:    cmpi.b   #$1b,d0               'ESC'-Taste?
307:    beq.s    neuadr                ja, neue Adresse
308:
309:    swap     d0                    sonst 'Cursor-down'..
310:    cmpi.b   #$50,d0               ..Taste?
311:    bne.s    weiter
312:
313:  allrght    bsr      scroll    Bildschirm scrollen,..
314:
315:    lea      zeile(pc),a5          ..neue Zeile ausgeben..
316:    bsr      txt_out
317:    bra      loop                  ..und wieder von vorne
318:
319:  *************** ENDE des ersten Teils **********
```

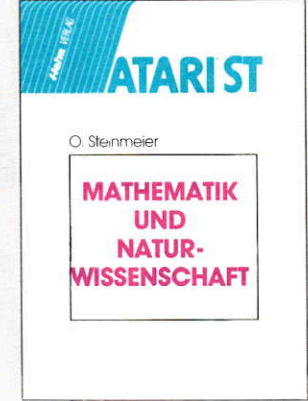
# NEU & AKTUELL




Buch incl. Programm-Diskette
Hardcover B-406
**DM 59,-**




Über 570 Seiten
mit Programmdisketten
Hardcover B-415
**DM 59,-**




ca. 300 Seiten
Hardcover B-409 DM 49,–
Programmdiskette
zum Buch: D-249 DM 39,–




über 530 Seiten
Bestell-Nr. B-419 54,–
ISBN 3-923250-69-X
Inclusive Programmdiskette



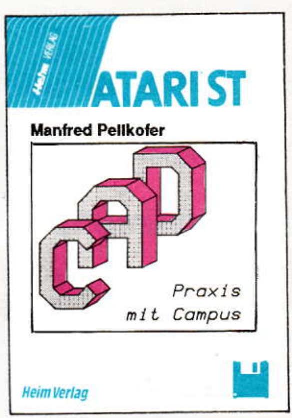

Über 300 Seiten
B-414
**DM 49,-**




Hardcover, über 430 Seiten
mit Programmdiskette
B-421
**DM 69,-**




Hardcover, 453 Seiten
mit Programmdiskette
B-400
**DM 59,-**




ca. 290 Seiten
Bestell-Nr. B-418 59,–
ISBN 3-923250-67-3
Inclusive Programmdiskette




Hardcover
Bestell-Nr. B-436 DM 59,-
ISBN 3-923250-77-0
Inclusive Diskette mit Interaktiver Assembler-Entwicklungssoftware




220 Seiten – Hardcover
Bestell-Nr. B-432 DM 49,-
ISBN 3-923250-76-2
Inclusive Diskette mit Sicherheitssystem




Hardcover
Bestell-Nr. B-435 DM 59,-
ISBN 3-923250-79-7




über 330 Seiten
Bestell-Nr. B-420 54,–
ISBN 3-923250-70-3

Alle Preise sind unverbindlich empfohlene Verkaufspreise.

## BESTELL-COUPON

an Heim-Verlag
Heidelberger Landstraße 194
6100 Darmstadt-Eberstadt

Bitte senden Sie mir: ______________________

zuzügl. Versandkosten DM 6,– (Ausland DM 10,–) unabhängig von der bestellten Stückzahl

☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name, Vorname ______________________

Straße, Hausnr. ______________________

PLZ, Ort ______________________

Benutzen Sie auch die in ST COMPUTER vorhandene Bestellkarte.




Halle 7 • Stand E46

**Heim Verlag**

*Heidelberger Landstraße 194*
*6100 Darmstadt-Eberstadt*
*Telefon 0 61 51 - 5 60 57*

**Schweiz**
Data Trade AG
Landstr. 1
CH - 5415 Rieden - Baden

**Österreich**
Haider
Computer + Peripherie
Grazer Str. 63
A - 2700 Wiener Neustadt

# Anwendungen in dBMAN

## Aller guten Dinge sind drei

**Nun also zum letzten Teil unserer dBMAN-Reihe, und damit zu den Ausgabeprozeduren in unserem kleinen Programm zur Stundenberechnung.**

### Die Konzeption

Wie bereits im letzten Teil angedeutet, kann aufgrund der gewählten Datenbank-bankstruktur und des Indexschlüssels die Auswertung sinnvollerweise nur nach dem Datum erfolgen. Dem Anwender sollen grundsätzlich drei Möglichkeiten zur Verfügung stehen, den Zeitraum zu definieren. Im Programm-Modul *V_SELLI.CMD* entscheidet er sich für

- die Auswertung nach einem expliziten Anfangs- und Enddatum
- die Auswertung für eine bestimmte Woche
- die Auswertung für einen bestimmten Monat

Dabei soll für die Wochen- und Monatsauswertung die Eingabe eines einzigen Datums genügen. Aufgrund dieses Datums werden dann das exakte Anfangs- und Enddatum des Zeitraumes berechnet. Anschließend erfolgt im Programm-Modul *KONDRU.CMD* die Auswahl der Ausgabeeinheit. Es stehen Bildschirm, Drucker oder Datei zur Verfügung. In diesem Modul werden dann je nach Bedarf die Druckeranpassung geladen, ein Dateiname für die Textdatei erfragt und gegebenenfalls eine bereits vorhandene Datei gleichen Namens gelöscht. Hier besteht dann für den Anwender die letzte Möglichkeit, 'es sich nochmal anders zu überlegen' und den Ausdruck zu stoppen, ansonsten wird das eigentliche Programm-Modul *V_ALI.CMD* aufgerufen. Dort erfolgt dann der Ausgabevorgang auf die gewählte Einheit (s. Abb. 1).

### Nochmal Variablen-Handling

Die oben beschriebene Verteilung der verschiedenen Aufgaben auf einzelne Programm-Module hat logischerweise zur Folge, daß einzelne Variablen über drei mit DO ... aufgerufene Programme 'durchgeschleift' werden müssen. Zusätzlich kommt im zweiten Modul eine je nach Ausgabeeinheit unterschiedliche Anzahl von Y.Variablen hinzu. Deshalb ist es sinnvoll, die Variablen für das

```
Anfangsdatum  -> BEG
das Enddatum  -> END
die Kopfzeile  -> HDG
die Bedingung  -> BED
```

zunächst als temporäre Variablen ohne Präfix zu definieren. An das zweite Modul werden sie als Parameter übergeben. Um sie auf die gleiche Ebene zu bringen wie die Variablen, die in diesem Modul hinzukommen, definieren wir sie als Y.Variablen. Als solche können sie im folgenden Modul als Z.Variablen weiterverarbeitet, und anschließend automatisch gelöscht werden (siehe auch Teil 1 dieser Reihe).

Zur Auswahl des Ausgabezeitraumes verwenden wir wieder ein vertikales Menü. Wie man sieht, kann man Erläuterungen zu den einzelnen Menüpunkten auch berechnen. Dies muß allerdings zuvor in einer eigenen Variablen geschehen, wenn bei komplexeren Berechnungen mehr als zwei verschiedene Anführungszeichen (' und ") nötig wären (s. Listing 1). Nur wenn es erforderlich ist *(VMENU()=1)*, wird ein zweites Datum angefordert.

Die Auswertung der Menüauswahl erfolgt in zwei Schritten. In der CASE-Auswertung wird erst einmal festgestellt, ob überhaupt ein Datensatz vorhanden ist, der der Bedingung entspricht. Die (programmtechnisch) einfachste Möglichkeit wäre, die Datenbank zu filtern. Dazu dient die Funktion *SET FILTER TO* Bedingung (L).

Dies ist aber(selten ein Nutzen, wo nicht ein Schaden...) zugleich auch die extrem langsame Lösung, da jedesmal die gesamte Datenbank durchsucht werden muß. Besonders bei großen Datenbanken führt dies zu erheblichen Wartezeiten. Bildhaft kann man sich das in etwa so vorstellen:

```
SET FILTER TO Bedingung entspricht
DELETE ALL FOR .NOT. Bedingung
SET DELETED ON
....
....
SET FILTER TO          entspricht
RECALL ALL FOR ...
SET DELETED OFF
```

Manuell ausgeführt müßte man, um eine Datenbank zu "filtern", alle Datensätze, die nicht den Filterbedingungen entsprechen, als "gelöscht" markieren und sie mit dem Befehl *SET DELETED ON* ausblenden. D.h. es wird so getan, als ob die ganze Datenbank nur aus den der Bedingung entsprechenden Datensätzen besteht. Existiert kein Datensatz, so gilt *EOF()=BOF()* oder richtiger *EOF() = T .AND. BOF() = T* (vgl. Teil 2 dieser Reihe). Eine Indexdatei, die die Selektion der Datensätze erleichtern und damit unabhängig von der Größe der Datenbank erheblich beschleunigen könnte, wird bei diesem Befehl nicht genutzt.

Unsere Indexdatei ist ja so angelegt, daß die Datensätze nach *DATUM* und *ANFANGSZEIT* sortiert erscheinen (vgl. Teil 2 dieser Reihe). Um einen bestimmten Zeitraum darzustellen, genügt es also, den ersten zutreffenden Datensatz mit *FIND* anzuspringen und solange einen Datensatz weiterzu"skippen", bis die Bedin-

```
STUNDEN-ABRECHNUNG für die 45. Woche

am:                    TÄTIGKEIT:             von:    bis:    STUNDEN:

Mittwoch   15.11.1989  Siesta                 08:00 - 12:00   4.0 Std.

Samstag    18.11.1989  Mittagessen            12:30 - 13:30   1.0 Std.
Samstag    18.11.1989  Kaffeetrinken          14:00 - 15:00   1.0 Std.

GESAMT STUNDEN     6.00 Std.

                              BETRAG        120.00 DM
```

*Abb. 1: Bildschirm- und Druckerausgabe gleichen sich wie ein Ei dem*

gung nicht mehr zutrifft oder das Dateiende erreicht ist. Wenn wir diese Methode bevorzugen, müssen wir aber zwei grundsätzlich unterschiedliche Fälle betrachten.

1. Es werden nur die Datensätze für einen einzigen Tag gesucht (BEG=END).

2. Es handelt sich um einen Zeitraum (BEG>END).

Im zweiten Fall kann es ja möglicherweise zu der Situation kommen, daß zwar Datensätze für den ganzen Zeitraum vorhanden sind, aber kein Datensatz für den ersten Tag des entsprechenden Zeitraums existiert.

Um zu verhindern, daß in diesem Fall die Suche mit *FOUND()=F* und *EOF()=T* endet, dient die Funktion *APPROX()*. Sie bewirkt, daß, wenn kein entsprechender Datensatz gefunden wird, immer der nächsthöhere angesprungen wird. (bei absteigendem Index entsprechend der nächstniederigere), und für *FOUND()* immer T(rue) übergeben wird, falls die Datenbank nicht leer ist.

In der Standardkonfiguration ist *APPROX()* aus diesem (guten) Grund ausgeschaltet, und man sollte nach *ASSIGN APPROX(T)* tunlichst darauf achten, daß dieser Zustand auch sobald wie möglich wieder hergestellt wird. Bei sehr komplexen Programmen und besonders, wenn ein Rücksprung zum Hauptmenü aus allen Ebenen des Programms möglich ist, ist es sinnvoll, den APPROX-Schalter routinemäßig zu überprüfen. Dies geschieht mit

```
IF APPROX()=T .AND.
......
ASSIGN APPROX(F)
ENDIF
```

In unserer Anwendung wird *APPROX* sofort nach der CASE-Auswertung wieder ausgeschaltet; wenn *BEG=END* ist, muß dies bereits vor der Suche geschehen.

Ich denke, der Unterschied zwischen der *FIND-* (Zeitpunkt) und der *FILTER*-Bedingung *BED* (Zeitraum) ist klar: FIND ist der 1.Tag des Suchzeitraumes als C-STRING im Indexformat. Wenn der Schalter *EXACT* auf *OFF* steht (*SET EXACT OFF*), interessiert sich dBMAN nicht dafür, ob auch die Uhrzeit, die zwar zum Indexschlüssel gehört, nicht aber im *FIND*-Ausdruck enthalten ist, übereinstimmt. Die Variable *BED* ist logisch wahr (T), solange das Datum des jeweiligen Datensatzes im Suchzeitraum liegt.

Das Suchdatum für den ersten Tag einer Woche kann mit *DOW()* berechnet werden (vgl. Teil 1 dieser Reihe). Um uns eine ganze Reihe von String- und Datumsumwandlungen zu sparen, bestimmen wir den ersten Tag des Monats, indem wir einfach vom eingegebenen Datum zurückrechnen.

Die Variable *HDG* enthält, wie schon erwähnt, das Anfangs- und Enddatum bzw. die Woche oder den Monat für die Titelzeile des Ausdrucks. Auch sie wird "individuell" aufgrund der Menüauswahl zusammengestellt. In diesem Zusammenhang ist die Funktion *WOY(datum1, datum2)* [W(eek) O(f) Y(ear)] interessant. Sie übergibt die Anzahl der Wochen zwischen *Datum1* und *Datum2* als numerischen Wert. Der ist per Definition immer eine positive ganze Zahl. Besonders in kaufmännischen Bereichen wird sehr oft in Wochen gerechnet, z.B. *"Lieferung in der 36. Woche..."*. Ohne diese Funktion ist die automatische Berechnung von Wochen sehr mühsam und aufwendig.

Bei der Ausgabe für eine volle Woche soll dies in der Titelzeile angegeben werden. Zu berechnen ist also, welcher Woche des jeweiligen Jahres das Anfangsdatum entspricht. Dazu wählen wir als erstes Datum den 1.Januar des entsprechenden Jahres, ermitteln die Anzahl der Wochen und wandeln den Wert in einen zweispaltigen C-String um. Da das gesamte Jahr etwas mehr als 52 Wochen enthält, muß man sich je nach Anwendung entscheiden, ob diese eine Teilwoche als 53. Woche behandelt werden soll, oder ob man besser mit der Woche 0 beginnt, und es bei den üblichen 52 Wochen beläßt. In diesem Fall müßte von dem errechneten Wert 1 subtrahiert werden.

## Weiter im Programm

Da wir ja an dieser Stelle des Programms noch nicht hundertprozentig sicher sein können, daß wirklich ein Datensatz gefunden wurde, der allen Bedingungen entspricht (APPROX!), muß noch eine endgültig Überprüfung erfolgen, bevor alle relevanten Variablen an das nächste Programm übergeben werden können. Im Prinzip wird in drei Schritten überprüft:

*? FOUND()* überprüft eigentlich nur den Fall *BEG=END*, da dann APPROX(F) gilt.

*? EOF()* überprüft *VMENU()=1* und *BEG<END*

*SAMEWEEK(datum1,datum2)* und *SAMEMONTH(datum1,datum2)* übergeben den logischen Wert "T", wenn beide Daten zu derselben/demselben Woche/ Monat gehören.

Sie ermöglichen die Überprüfung "in einem Aufwasch", wobei man jedoch penibel auf die "Klammerung" achten muß. Übersteht der gefundene Datensatz auch diese Tortur, können wir getrost zur Vorbereitung der Ausgabe übergehen.

## Druck

Wie bereits in der Einleitung kurz dargestellt, erfolgt nun in einem eigenen Programm-Modul die Wahl der Ausgabeeinheit. Das ist deshalb sinnvoll, weil dieses Modul auch für weitere Ausgaben verwendet werden kann. Selbstverständlich erfordert dies etwas Disziplin, da man sich wenigstens an seine eigenen Konventionen halten muß. Die übergebenen Parameter wie *ANF* oder *END* können ja statt eines Datums genauso gut eine Kundennummer oder in unserem Fall zusätzlich einen *CODE* für die Art der ausgeführten Tätigkeit enthalten.

Das Datenbanksystem dBMAN bietet dem Programmierer eine ganze Reihe von Möglichkeiten, Drucker anzusteuern. Die eingebauten *SET*-Befehle funktionieren

```
.LIST FY MEMORY

Y.LOCAL

ADV     C  <27><102><49>
BRAUS   C  <27>W<0>
BREIN   C  <27>W<1>
CON     C  <27><15>
CPI10   C  <27><80>
CPI12   C  <27><77>
CPI15   C  <27><103>
EL      C  <27><107><19>
FA      C  <27>F
FE      C  <27>E
HEL     C  <27><107><16>
HOAUS   C  <28>E<0>
HOEIN   C  <28>E<1>
INIT    C  <27><64>
ITC     C  <27><107><15>
ITEIN   C  <27><34>
LPI6    C  <27><50>
LPI8    C  <27><48>
LQAUS   C  <28>X<0>
LQEIN   C  <28>X<1>
LSV     C  <28><51>
SKIP0   C  <27><79>
SKIP5   C  <27><78><5>
STYLE   C  <27>!
SUAUS   C  <27>T
SUB     C  <27>S<1>
SUP     C  <27>S<0>
TIAUS   C  <27><35>
TIMES   C  <27><107><18>
UA      C  <27>-<1>
UE      C  <27>-<0>
Variable count : 35
```

*Abb. 2: Beispiel einer (im Dialog) erstellten MEMORY-Datei DRUCKER.MEM für NEC P6+*

auf (fast) allen Druckern. Will man jedoch anspruchsvollere Anwendungen programmieren, erlaubt es DBMAN durch den Befehl *?? ESC(var)*, alle ESC-Sequenzen eines Druckers anzusprechen.

Es gibt sicherlich verschiedene "Tricks" diese Sequenzen mehr oder weniger flexibel zu gestalten. Denn wenn der Anwender einen neuen, (nicht abwärts kompatiblen) Drucker betreiben will, kann die Anpassung an den neuen Drucker u.U. recht aufwendig werden. In jedem Fall sollten die ESC-Sequenzen außerhalb des Programms gespeichert werden, um sie gegebenenfalls separat und unabhängig von einem möglicherweise bereits compilierten Programm ändern zu können.

Die Variablen für die Druckersteuerung müssen folgender Syntax entsprechen, damit eine spätere Anpassung vorgenommen werden kann. [ Ein kleines (aber feines) Programm, mit dem auch ohne Programmierkenntnisse diese Sequenzen durch den Anwender selbst geändert werden können, erhalten Interessierte bei mir.] Alle variablen ESC-Sequenzen müssen in der Form *VARIABLENNAME* = *'<nnn><nnn><nnn>'* gespeichert werden (s. auch Abb. 2). nnn ist dabei die jeweilige Druckersequenz in dezimaler Form. ESC-Sequenzen, die mehr als drei Einträge besitzen, wie z.B. die Sequenz zum Leeren des Druckerpuffers und zum anschließenden Zeilenvorschub um n Zeilen beim NEC P6-plus (27,102,49,n), werden dann einfach in der Form *Y. variablenname* +'*<n>*' geschrieben, wobei <n> wiederum eine Variable oder ein fester Wert sein kann (s. Listing 2). Die benötigten ESC-Sequenzen müssen mit *SAVE X,Y,Z TO Laufwerk/Pfad DRUCKERNAME.MEM* gespeichert werden. Vor dem Drucken muß man die Sequenzen durch *RESTORE FX,Y,Z FROM DRUCKER.MEM ADDITIVE* laden. Der Parameter *ADDITIVE* bewirkt, daß bereits vorhandene Variablen auf der gleichen Ebene nicht gelöscht werden. Allerdings werden Variablen gleichen Namens durch den RESTORE-Wert überschrieben (s. Listing 3).

```
 1: * PROCEDURE SELLI
 2: * AUSWAHLMENÜ SELEKTIONSKRITERIEN
 3:
 4: * VARIABLEN DEFINITION
 5: ANF = DATE()-7
 6: END = DATE()
 7:
 8: * MENÜ DEFINITION MIT HILFSTEXTEN
 9:     X.KM1='DATUM%!z.B. vom '+DTOC(ANF)+'
              bis tt.mm.jjjj'
10:     X.KM2='WOCHE%!alle Tage derselben Woche'
11:     X.KM3='MONAT%!alle Tage desselben Monats'
12:     X.KM4='        '
13:     X.KM5=' EXIT     '
14: SET MESSAGE FIELD TO 24,5,40
15:
16: * MENÜ
17: @ 4,2 TO 14,15
18: ASSIGN VMENU(",",X.KM1,5,1,0,1,3,X.KM2,X.KM3,X.KM4,X.KM5)
19: @ 4,2 CLEAR TO 14,15
20:
21: * EXIT ?
22: IF VMENU() >3
23:     RETURN
24: ENDIF
25:
26: * MENÜ EINGABE
27: @ 18,25 SAY 'von ' GET ANF
28: IF VMENU() = 1
29:    @ 19,25 SAY 'bis ' GET END VALID END >= ANF
30: ENDIF
31: READ
32:
33: * DEFINITION  SUCHMODUS, SCHLÜSSELWORT, TITELZEILE
34: ASSIGN APPROX(T)
35: DO CASE
36: CASE VMENU() = 1
37:      HDG = 'vom  '+DTOC(ANF)+' - '+DTOC(END)
38:      BED = 'TAG <= Z.END'
39:      IF ANF = END
40:         HDG = 'vom  '+DTOC(ANF)
41:         ASSIGN APPROX(F)
42:      ENDIF
43:      FIND DTOK(ANF)
44: CASE VMENU() = 2
45:      FIND DTOK(ANF-(DOW(ANF)-1) )
46:      BED = 'SAMEWEEK(TAG,ANF)'
47:      HDG = 'für die '+ STR(WOY(CTOD('01.01.'
                      +STR(YEAR(ANF),4,0)),ANF),2,0)+;
48:      '.  Woche '
49: CASE VMENU() = 3
50:      FIND DTOK(ANF-DAY(ANF)+1)
51:      BED = 'SAMEMONTH(TAG,ANF)'
52:      DAT = ANF
53:      HDG = 'für '+&MO
54: ENDCASE
55: ASSIGN APPROX(F)
56: * ENDGÜLTIGE SELEKTION
57: IF .NOT. EOF() .AND. (FOUND() .AND.(SAMEWEEK(TAG,ANF) .AND. VME
                                                         NU()=2 ) ;
58:    .OR.   (SAMEMONTH(TAG,ANF) .AND.VMENU()=3) .OR. VMENU()=1)
59:          HDG = 'STUNDEN-ABRECHNUNG '+HDG
60:          *  ZUR DRUCKER UND EINHEITEN DEFINITON mit PARAMETERN
61:          DO V_KONDR WITH HDG,BED,ANF,END
62: ENDIF
63: RETURN
```

*Listing 1: V_SELLI.CMD*

dBMAN würde es also ermöglichen, über die Fileselectbox aus einer Reihe unterschiedlicher "Druckertreiber" auszuwählen, falls mehrere Druckertypen zur Verfügung stehen. Ein weiterer Vorteil besteht darin, daß ein Variablenname anschaulicher Auskunft über die Auswirkung der jeweiligen Sequenz gibt als irgendwelche Zahlenreihen.

Z.B. bedeutet *Y.LPI18* und *Y.CPI12* folgerichtig: Zeilenabstand auf 18 Zeilen pro Inch (Lines Per Inch) und die Zeichensatzgröße auf 12 Buchstaben pro Inch (Characters per Inch) einstellen (vgl. auch Abb. 2).

Die Variable *Y.PRINT* zeigt dem eigentlichen Druckprogramm an, ob die ESC-Sequenzen zur Verfügung stehen, denn bei der Ausgabe in eine Datei, die später mit einer Textverarbeitung weiterverarbeitet werden soll, erzeugen die Druckersteuerzeichen nur unnötige Symbole und Leerzeilen.

Die Umleitung des Drucks in eine Datei erfolgt mit dem Befehl *SET PRINTER/ ALTERNATE TO Device Nummer/Dateiname* und wird mit dem *SET PRINT/ ALTERNATE ON/OFF* für *?* und *??* und mit *SET DEVICE TO PRINT/SCREEN* für *@ ROW()-,COL()-* und *SAY*-Anweisungen aktiviert bzw. deaktiviert. Prinzipiell kann man also die ESC-Sequenzen auch einfach durch *SET PRINT OFF* ausblenden und mit *PRNFLG()* die Stellung des *SET*-Schalters ermitteln.

Die Eingabe des Dateinamens für die anzulegende Textdatei muß daraufhin überprüft werden, ob er nicht erlaubte Leerzeichen enthält, die mit den *TRIM*-Funktionen nicht entfernt werden und bei der Ausführung der Zeile 48 einen SYNTAX-Error erzeugen würden. Andere nicht erlaubte Zeichen wie Punkt, Fragezeichen etc. werden durch die *PICTURE*-Maske verhindert. Laufwerke und File-Extensions (.TXT) kann man dadurch auch nicht manuell eingeben.

Falls schon eine Datei gleichen Namens existiert, wird sie vorher (ohne Warnung und ohne Rücksicht auf Verluste) gelöscht, da die Ausgabe andernfalls einfach an die bestehende angehängt wird.

Falls die Ausgabe auf dem Bildschirm erfolgen soll, sind hier keine besonderen Einstellungen erforderlich (vgl. Abb. 1). Die nun folgende eigentliche Ausgabe soll die zutreffenden Datensätze auflisten, die jeweils angefallenen Stunden addieren und im Fall der Ausgabe auf den

```
 1: * PROCEDURE V_ALI
 2: * AUSGABE FÜR ALLE EINHEITEN
 3: ERASE
 4: Y.STDN = 0
 5: X.STDL = 25.00
 6: * KOPFZEILE
 7: Y.HDG0 = 'am:'+SPACE(21)+'TÄTIGKEIT:'+SPACE(18            )+'von:
                                                        bis:     STUNDEN:'
 8: @ 1,5 SAY Z.HDG
 9: @ 2,0 TO 2,80
10: @ 3,1 SAY Y.HDG0
11: @ 4,0 TO 4,80
12:
13: * CHARACTER FÜR DIE LINIE ANPASSEN
14:     IF Z.PRINT
15:     @ ROW()+1,0 SAY DUPCHAR(IIFN(Z.FNAME<>' ',45,196),86)
16:     ENDIF
17:
18: * LISTE
19: DO WHILE .NOT. EOF() .AND. &Z.BED
20: DAT = ABRV->TAG
21:
22: @ ROW()+1,1 SAY &WOT
23: @ ROW()   ,13 SAY  DTOC(ABRV->TAG)
24: @ ROW()   ,25 SAY TRIM(ABRV->ACT)
25: @ ROW()   ,52 SAY '          '+ABRV->VON
26: @ ROW()   ,60 SAY '- '+ ABRV->BIS
27: @ ROW()   ,70 SAY STR(TIMETON(TIMESUB(ABRV->BIS
                ,ABRV->VON),2),4,1)+ ' Std.'
28:
29: Y.STDN = Y.STDN+TIMETON(TIMESUB(ABRV->BIS,ABRV->VON),2)
30: Y.VAR = ABRV->TAG
31: SKIP
32:
33: * LEERZEILE FÜR NEUEN TAG
34:     IF ABRV->TAG <> Y.VAR
35:        @ ROW()+1,0 SAY ' '
36:     ENDIF
37:
38: * BILDSCHIRM VOLL
39: IF ROW() > 20 .AND. Z.PRINT = F
40:     @ ROW()+1,0 TO ROW(),80
41:     @ ROW()+1   ,40 SAY 'GESAMT STD     '+ STR(Y.STDN,5,2)+' Std. '
42:     WAIT
43:     @ 5,0 ERASE
44:     ASSIGN ROW(4)
45: ENDIF
46: ENDDO
47:
48:
49: * FUPZEILE
50:     IF Z.PRINT
51:     @ ROW()+1,0 SAY DUPCHAR(IIFN(Z.FNAME<>' ',45,196),86)
52:     ENDIF
53: @ ROW()+1,0 TO ROW(),80
54: @ ROW()+2   ,5 SAY 'GESAMT STUNDEN     '+ STR(Y.STDN,5,2)+' Std. '
55: * AUSGABE BEENDEN
56:     IF  Z.PRINT
57:         SET PRINT TO 0
58:         SET DEVICE TO SCREEN
59:         IF Z.FNAME = '          '
60:            ?? ESC(Z.ADV+'2')
61:         ENDIF
62:         SET PRINT OFF
63:     ELSE
64:         @ ROW()+1,0 TO ROW(),80
65:         @ ROW()+1   ,40 SAY 'BETRAG      '+STR(Y.STDN*X.STDL,8,2)+ ' DM'
66:          @ ROW()+1,0 TO ROW(),80
67:         WAIT
68:     ENDIF
69: RETURN
```

*Listing 2: V_ALI.CMD*

Bildschirm mit einem bestimmten Stundenlohn multiplizieren.

Da die Ausgabe ja für alle Einheiten funktionieren soll, sind ein Paar Dinge zu beachten. Die mit dem *@ zeile,spalte TO zeile,spalte* gezogenen Linien beziehen sich lediglich auf die Ausgabe auf dem Bildschirm, da sie kein ? bzw. kein *SAY* enthalten. Für die Ausgabe der Linien auf den Drucker bzw. in die Datei müssen wir eigene Zeichen definieren. Wenn schon, denn schon...

Letztendlich ist es Geschmackssache, wie man die Linien aufs Papier bringt. Ich würde es in jedem Fall bevorzugen, für die Druckerausgabe die entsprechenden

IBM-Grafikzeichen zu verwenden, um schöne dünne durchgezogene Linien zu erhalten. Da die aber in einer Textdatei nur Unsinn erzeugen würden, können wir mit der bereits bekannten IFFN-Funktion den jeweils richtigen ASCII-Wert einsetzen (DEZ=196 für eine einfache durchgezogene Linie bzw. DEZ=45 für den normalen Bindestrich -"Schreibmaschinenlinie"). Die Funktion *DUPCHAR (ASCII-wert,Anzahl)* erzeugt dann die Linie bzw. den Strich.

Insbesondere für Ausgaben, deren Spaltenbreite je nach Inhalt der Felder verschieden ist, kann man mit dieser Funktion Linien an die jeweilige Breite anpassen, indem man den Parameter 'Anzahl' als Variable behandelt. Auf die elegantere Möglichkeit, auch den numerischen Wert des ASCII-Codes bereits im Programm-Modul *KONDR.CMD* als Variable zu definieren bzw. den kompletten Ausdruck als MACRO zu übernehmen, habe ich dem 'roten Faden' zuliebe verzichtet, obwohl man sich zumindest die IFF-Funktion dadurch erspart.

Ein weiterer beachtenswerter Punkt, der sich durch die unterschiedlichen Ausgabeeinheiten ergibt, ist der Seitenumbruch, der ja bei der Bildschirmausgabe bereits bei Zeile 20 erfolgen muß. Nach der Ausgabe einer Zwischensumme muß dann der Bildschirm ab der Kopfzeile gelöscht werden.

Da die Ausgabe mit *ROW()+1* fortgesetzt wird, sorgt *ASSIGN ROW(n)* für die Fortsetzung ohne Leerzeile, falls nicht ohnehin schon das Dateiende erreicht ist. Wenn keine bestimmte Zeile eingestellt werden soll, kann man auch *ASSIGN ROW(ROW()-1)* schreiben. Funktionen dieser Art funktionieren natürlich nur auf dem Bildschirm. Beim Ausdruck würde dieser Befehl bei den meisten Druckern einen Seitenvorschub auslösen.

In unserem Beispiel überlassen wir das Problem des Seitenvorschubes beim Ausdruck aufs Papier besser dem Drucker (SKIP=ON), obwohl dBMAN mit einer ganzen Reihe von *SET*-Befehlen eine äußerst komfortable Einstellung der Ränder bzw. Seitenlänge, aber auch die Definition von mehreren Kopf- und Fußzeilen erlaubt. Effektiv können Sie aber nur in einem eigenen Druckprogramm-Modul eingesetzt werden, wenn nicht zusätzlich eine Ausgabemöglichkeit auf dem Bildschirm [illegible] soll.

```
 1: * PROCEDURE V_KONDR
 2: * DRUCKER UND EINHEITEN DEFINITON
 3:
 4: * PARAMETER ÜBERGABE
 5: PARAMETER HDG,BED,ANF,END
 6: *
 7: * MENÜ DEFINITION
 8: Y.AMEN1 = 'BILDSCHIRM'
 9: Y.AMEN2 = 'ASCII-FILE'
10: Y.AMEN3 = 'DRUCKER I '
11:
12: * VARIABLEN DEFINITION
13: Y.HDG   = HDG
14: Y.BED   = BED
15: Y.ANF   = ANF
16: Y.END   = END
17: Y.FNAME = SPACE(8)
18: Y.PRINT = F
19:
20: * MENÜ
21: @ 4,2 TO 12,15
22: ASSIGN VMENU(',',Y.AMEN1,5,1,0,1,3,Y.AMEN2,Y.AMEN3)
23:
24:    IF VMENU() = 3
25:       * DRUCKER KONFIGURIEN
26:       RESTORE FY FROM NEC_P6.MEM ADDITIVE
27:       SET LEFT MARGIN TO 2
28:       SET TOPFORM
29:       SET PRINT TO 0
30:       SET PRINT ON
31:           ?? ESC('<27><107><0>')
32:           ?? ESC(Y.CPI12)
33:           ?? ESC(Y.LPI8)
34:       Y.PRINT = T
35:       SET DEVICE TO PRINT
36:   ENDIF
37:   IF VMENU() = 2
38:         * ASCII AUSGABE VORBEREITEN DATEINAME
39:         @ 18,25 ESAY 'DATEINAMEN : '
40:         @ 19,24 ESAY ' ' GET Y.FNAME PICT'AAAAAAAA' VALID
                         Y.FNAME <> '    ';.AND. AT(' ',LTRIM
                         (TRIM(Y.FNAME)))<1
42:                READ
43:               Y.FNAME=LTRIM(TRIM(Y.FNAME))
44:           IF FILE('A:\&Y.FNAME..TXT')
45:               DELE FILE A:\&Y.FNAME..TXT
46:         ENDIF
47:         * AUSGABE UMLENKEN
48:         SET PRINT TO A:\&Y.FNAME..TXT
49:         Y.PRINT = T
50:         SET DEVICE TO PRINT
51:    ENDIF
52: * EIGENTLICHES DRUCKPROGRAMM AUFRUFEN
53: SET APPROX OFF
54: DO V_ALI
55: RETURN
```

*Listing 3: V_KONDR.CMD*

Auch in diesem Fall bestünde natürlich die Möglichkeit, die Zeilennummer bereits im Modul *V_KONDR.CMD* als Variable *(Y.PGLEN = 20 bzw 60)* zu definieren. Wenn man dann den Bildschirm vollständig mit ERASE löscht und die Kopfzeilen nochmal schreibt, erhält man bei der Druckerausgabe automatisch einen Seitenvorschub und neue Kopfzeilen.

Als Fußzeile wird die Gesamtanzahl der Stunden angezeigt und im Fall der Bildschirmausgabe mit einem (festzusetzenden) Stundenlohn multipliziert. Die Stunden werden in der Variablen *Y.STDN* als numerischer Wert aufsummiert.

Um die Übersichtlichkeit der Liste zu verbessern, werden zwischen den einzelnen Tagen Leerzeilen eingefügt. Dazu wird das Datum des aktuellen Datensatzes in der Variablen *Y.VAR* zwischengespeichert, mit dem des nächsten Datensatzes verglichen und im Fall eines neuen Tages eine Leerzeile eingefügt.

Auch ein einzelnes Fragezeichen '?' würde einen Zeilenvorschub auslösen, deshalb müssen ESC-Sequenzen immer mit zwei Fragezeichen eingeleitet werden. Die Zeilen 56-62 im Listing 2 stellen die Standardkonfiguration wieder her.

## Prozeduren

Die einzelnen Module sind so aufgebaut, daß sie sich ganz einfach zu einer Prozedur zusammenfügen lassen. Es müssen nur die Kommentarsternchen '*' vor den Zeilen 1 entfernt werden. Nun kann das gesamte Programm mit *>SET PROCEDURE TO* Dateinameund *DO V_ALI* z.B von einem übergeordneten Menü aus gestartet werden. Die Dateierweiterung *.CMD bei den Programmaufrufen läßt man am besten ganz weg, da man sich sonst ein für allemal entweder für *.RUN oder *.CMD entscheiden müßte. Zwei Dinge sind aber prinzipiell bei dem Arbeiten mit Prozeduren zu beachten.

Mit dem Öffnen einer Prozedur legt dBMAN die sogenannte "PROCEDURE-TABLE" an. Darin sind die "Sprungadressen" aller Programm-Module der jeweiligen Prozedur enthalten. Solange man seine Prozedur nicht compiliert hat, legt dBMAN diese Informationen in einer Datei mit dem Prozedurnamen und der Erweiterung *.PRO in dem unter *SET DEFAULT SYSTEM TO* definierten Pfad ab.

Da dBMAN die Zeilen in der Prozedur nicht relativ zum Dateianfang, sondern relativ zur ersten Zeile des jeweiligen Moduls zählt, können die DEBUG-Kommandos *REPLACE CMD-LINE* etc. beim Edieren von Prozeduren nicht angewandt werden. Es können nämlich nach dem Zusammenlinken der Module zu einer Prozedur Fehler auftreten, die bei Einzelmodulen keine *ERRMSG* erzeugen. Zum Beispiel ein fehlendes *RETURN* am Ende eines Modules macht bei Einzelmodulen keine Probleme. Erst in der Prozedur wird durch das fehlende *RETURN* das Modulende nicht mehr erkannt, und DBMAN erzeugt eine fehlerhafte Procedur-Table.

Ändert man nun den Quelltext einer Prozedur in der Weise, daß eine Zeile eingefügt oder gelöscht wird, erzeugt dBMAN nur dann eine neue Procedur-Table, wenn man die alte vorher gelöscht hat. Ansonsten würde die alte Table benutzt und dadurch falsche Zeilen angesprungen. Bei der compilierten Prozedur wird die Procedur-Table bei jedem Aufruf der Prozedur neu erstellt.

*Peter Neuchel*

# Per SCSI zum ST

## TEIL III: Die Software

**Endlich kommen die Software-Spezialisten zum Zuge, die nur darauf warten, die Welt der Massenspeicher durch neuartige Dienstprogramme zu beleben. Nach der Beschreibung der programmtechnischen Möglichkeiten, die dieser Host-Adapter bietet, folgt noch einmal ein abschließender Blick auf die Hardware. Dabei geht es um Sonderfälle, Verbesserungen und eventuelle Unklarheiten.**

Den Anfang bildet aber ein Beispiel, wie überhaupt auf den DMA-Chip zugegriffen werden muß, um eine gewünschte Übertragung zu erreichen. Das Assembler-Programm (Listing 1) enthält bereits alle Routinen, die für einen einfachen Zugriff notwendig sind. In diesem Fall geht es um Kommandos, die einige Informations-Bytes liefern, wie z.B. *REQUEST SENSE*, *INQUIRY* oder *READ CAPACITY*. Der Quell-Code ist zum großen Teil der 'Kleisterscheibe' entnommen. Ich bitte um freundliche Beachtung des Programmkopfes.

Listing1 beginnt mit der Definition einiger globaler Variablen, durch die ein Hochsprachenprogramm mit diesen hardwarenahen Funktionen kommunizieren kann. Ich habe diesen Weg der Parameterübergabe gewählt, um weitgehend unabhängig von den Eigenheiten verschiedener Compiler zu bleiben.

### Der ganz normale Wahnsinn

Zunächst sind einige Vorbereitungen nötig, wie das Umschalten in den Supervisor-Modus und das Sperren des Floppy-Interrupts. Nachdem die Adresse des Kommandoblocks geladen wurde, wird der Durchlaufzähler eingestellt. Wozu denn das? Der DMA-Chip hat die Eigenart, gelesene Daten nur in 16-Byte-Päckchen abzuliefern. Liefert ein Kommando weniger Daten bzw. eine nicht durch 16 teilbare Anzahl, so gehen womöglich wichtige Daten verloren. Die Lösung ist aber sehr einfach: Das Kommando wird so oft wiederholt, bis sichergestellt ist, daß auch alle Daten angekommen sind. Die Anzahl der Wiederholungen bestimmt der Durchlaufzähler.

Durch 'Toggeln' eines Bits im Modusregister wird alte Information gelöscht, und danach erhält das Access-Register die Anzahl der zu übertragenden Sektoren. In unserem Fall reicht ein Sektor völlig aus. Nun muß der Chip noch wissen, wo er denn all die Daten hinterlegen soll. Dazu wird die Pufferadresse byteweise in den Adreßzähler geladen.

Daß die Länge des Kommandoblocks übergeben werden muß, stellt bereits eine Erweiterung dar. Festplattenkommandos der Gruppe 0 besitzen nämlich grundsätzlich einen 6-Byte-Kommandoblock. Alle zusätzlichen Kommandos benutzen, wie teilweise auch das CD-ROM, 10-Byte-Blöcke.

Alle bisherigen Befehle dienten der Vorbereitung. Mit dem Label *'loop:'* beginnt die eigentliche Ausführung. Überlesen Sie bitte zunächst den Aufruf von *'setgroup'* zur Kommandogruppenwahl - die wird erst später benötigt. Prinzipiell geschieht nun folgendes: Die Kommando-Bytes werden zusammen mit anderen Informationen nacheinander zum Controller geschickt. Nach jedem Byte wird auf eine Reaktion des angeschlossenen Gerätes gewartet, indem ein Bit im Gpip-Register getestet wird. Erfolgt keine Reaktion, weil beispielsweise gar nichts angeschlossen ist, wird nach einer vorgegebenen Zeit *'Timeout'* gemeldet. Nach dem letzten Kommando-Byte sollte man etwas länger warten, damit die Übertragung abgeschlossen werden kann. Ist bis hierhin alles fehlerfrei gelaufen, wird das Status-Byte gelesen und damit die Übertragung beendet. Eine Kombination aus DMA- und SCSI-Status wird der aufrufenden Funktion in D0 als long-Wert übergeben. Die ordentliche Kommentierung des Listings macht es sicher möglich, den Ablauf detaillierter nachzuvollziehen.

### Do you speak C ?

Das kurze C-Programm (Listing 2) entlockt einem SCSI-Gerät mit dem INQUIRY-Befehl Informationen über Name, Typ, Version und so weiter. Die Funktion SCSI() und der Datenpuffer werden als 'extern' deklariert, da sie im Assembler-Modul realisiert sind. Von den drei übrigen globalen Variablen schauen wir uns zunächst nur zwei an, nämlich *Comlen* und *Comblock*. Die Initialisierung erfolgt der Einfachheit halber gleich bei der Definition. INQUIRY ist ein einfaches Gruppe-0-Kommando, begnügt sich also mit einem sechs Byte großen Kommandoblock. Davon werden nur das erste Byte als Opcode ($12) und das fünfte Byte als Übertragungslänge verwendet. Dem ersten Kommando-Byte ist zu entnehmen, daß das Beispiel für die ACSI-Adresse 0 ausgelegt ist. Bei Adresse 2 würde $52 im ersten Byte stehen.

Die Ausführung beschränkt sich nun darauf, die Funktion SCSI() aufzurufen und den Rückgabewert (long) als Status zu

interpretieren. Die Werte '1' und '3' zeugen von fehlerfreier Übertragung. In *'Buffer'* findet sich ab der Position 8 die gewünschte Information. Wollen Sie dieses Beispiel nachvollziehen, müssen Sie dem Linker explizit mitteilen, welche Objektprogramme (Endung .O) miteinander verbunden werden sollen. In diesem Fall sind es mindestens zwei, nämlich (z.B.) INQUIRY.O und DMAROUT.O. Dies ist deshalb erwähnenswert, da ein Aufteilen des Quell-Codes in Module und das 'aufwendigere' Linken wohl noch nicht die gewünschte Verbreitung gefunden hat. Man denke nur an die Bemühungen, den in Turbo-C fehlenden Inline-Assembler nachzubilden.

## Ist doch logisch

Richten Sie Ihren Blick doch noch einmal auf die Assembler-Routinen. Bei genauer Betrachtung der Daten, die der arme DMA-Chip zugeführt bekommt, ergibt sich zwangsläufig die in Bild 1 dargestellte Folge. Zunächst gelangt $88 in Wortlänge nach dmodus, danach eine Reihe Langwörter nach daccess, wobei das niederwertige Wort natürlich wieder in dmodus landet. Wie unschwer zu erkennen ist, handelt es sich um das Inquiry-Beispiel.

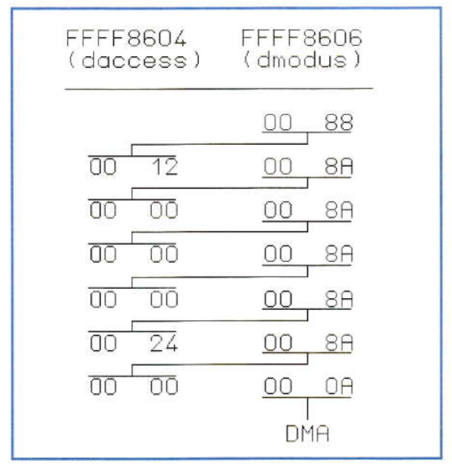



*Bild 1: Kommandoübertragung*

Der erste dmodus-Eintrag ($88) bedeutet ungefähr folgendes: Die höherwertige 8 läßt den DMA-Chip auf die Floppy schauen, damit er uns nicht durchgeht. Die Wirkung ist mit der von Scheuklappen durchaus vergleichbar. An der zweiten 8 fällt im Gegensatz zu den folgenden Einträgen nur auf, daß Bit 1 nicht gesetzt ist. Dieses Bit führt direkt zur A1-Leitung, und wenn diese auf logisch Null gesetzt, d.h. aktiviert ist, warten alle ACSI-Geräte auf das erste Kommando-Byte. Bei der folgenden Langwortübertragung wird zunächst das Kommando-Byte $12 ins daccess-Register übertragen, und erst danach wird dmodus mit $8A geladen. Dieser Wert bewirkt aber bereits eine Voreinstellung für das zweite Kommando-Byte, diesmal mit gesetztem A1-Bit. Darauf sind selbst die Atari-Entwickler bei ihrem TOS 1.0 noch hereingefallen, indem sie durch Programmierung von $88 auch dem zweiten Kommando-Byte ein aktiviertes A1-Bit mit auf den Weg gaben. Darunter leiden die ACSI-Geräte noch heute - auch dieser Host-Adapter, wie wir noch sehen werden.

Die nächsten vier Langwörter werden nach dem gleichen Prinzip übertragen. Mit dem letzten Kommando-Byte, welches üblicherweise Null ist, wird dann das Byte $0A ins Modusregister geladen. Hier ist das Scheuklappen-Bit gelöscht, und der DMA-Gaul rennt los. Die Zusammengehörigkeit der Daten ist in Bild 1 durch Verbindungslinien dargestellt.

## Kommandogruppen: Aufzucht und Pflege

Bis hierhin gehört alles noch unter den Oberbegriff 'DMA-Standard', ob mit oder ohne ST-Host-Adapter. Die bisherigen Routinen sollten alle ACSI-Platten verstehen, die mit dem vorgeschlagenen Befehl zurechtkommen. Während aber die Programmierung der DMA-Schnittstelle ein recht komplexes Gebiet ist, benötigt die freie Wahl der Kommando-

```
 1: ****************************************************************
 2: * Routinen zur Programmierung des DMA-Chips          20.01.90
 3: * Programmskelett aus dem BHDI übernommen
 4: * (Brod's Harddisk-Interface / Scheibenkleister II)
 5: * Mit freundlicher Genehmigung des Autors
 6: *
 7: * Einzelne Programmteile gekürzt und verändert
 8: * Nur für "REQSENSE-ähnliche" Kommandos
 9: * Erweitert um Kommandogruppenwahl           Reiner Wiechert
10: * (C) MAXON Computer
11: ****************************************************************
12:
13: globl SCSI                            ; Parameter für Hauptprogramm
14: globl Comblock
15: globl Comlen
16: globl Buffer
17: globl Comgroup
18:
19:
20: vbl     = $43e                        ; Floppy-VBL-Sperre
21: daccess = $ffff8604                   ; DMA-Access-Register
22: dmodus  = $ffff8606                   ; DMA-Modus-Register
23: dlow    = $ffff860d                   ; DMA-Adreßzähler, Lowbyte
24: dmid    = $ffff860b                   ; DMA-Adreßzähler, Midbyte
25: dhigh   = $ffff8609                   ; DMA-Adreßzähler, Highbyte
26: hz_200  = $4ba                        ; 200-Hz-Zähler
27: gpip    = $fffffa01                   ; GPIP-Register des MFP
28:
29: ****************************************************************
30: * Deklaration in 'C':  long SCSI(void)
31: ****************************************************************
32: text
33:
34: SCSI:
35:     moveq.l #0,d0                     ; Userstack wird Supervisorstack
36:     bsr     super                     ; in den Supervisormodus
37:     lea     save_ssp,a0               ; Stackpointer retten
38:     move.l  d0,(a0)
39:
40:     st      vbl                       ; Floppy-VBL sperren
41:
42:     lea     Comblock,a0               ; Adresse von Comblock
43:     moveq.l #1,d2                     ; Zähler für 2 Durchläufe
44:
45:     move.w  #$98,dmodus               ; DMA-Puffer und DMA-Status
46:     nop                               ; löschen
47:     move.w  #$198,dmodus
48:     nop
49:     move.w  #$98,dmodus
50:
51:     move.w  #1,daccess                ; Blockanzahl in Sektorzähler
52:
53:     move.l  #Buffer,d0                ; DMA-Startadresse setzen
54:     move.b  d0,dlow                   ; Lowbyte
55:     asr.l   #8,d0
56:     move.b  d0,dmid                   ; Midbyte
57:     asr.l   #8,d0
```

→

gruppe nur einen geringen Mehraufwand. Der Host-Adapter arbeitet nach dem Vorwahlprinzip. Unmittelbar bevor dem SCSI-Gerät ein Kommando übermittelt wird, kann die entsprechende Kommandogruppe eingestellt werden. Wird dies unterlassen, handelt es sich um ein Ortsgespräch, und die Gruppe 0 wird aktiviert. Das Unterprogramm *'setgroup'* am Ende von Listing 1 beinhaltet bereits alle notwendigen Aktionen. Die Variable *Comgroup* enthält das erste und einzige Kommando-Byte. Der wichtigste Aspekt dabei ist die Target-Nummer (ACSI-Adresse). Während die 'eigentliche' Adresse eine gerade Ziffer ist (0,2,4,6), wird die Kommandogruppe auf der folgenden ungeraden Nummer eingestellt. Übrigens: Sucht ein beliebiges Programm den ACSI-Bus nach angeschlossenen Geräten ab, wird auf dieser ungeraden Adresse ein Timeout erkannt. So soll es auch sein.

In der Variablen *Comgroup* ist auch die SCSI-ID verschlüsselt, mit der bis zu vier Controller ausgewählt werden können. Das Format kann man in Tabelle 1 sehen.

Ein Beispiel: Der Host-Adapter sei auf die Adresse 4 eingestellt. Nun soll ein Zugriff auf die zweite Harddisk (SCSI-ID = 1) mit einem Gruppe-2-Kommando erfolgen. In *Comgroup* steht somit '101 01 010' bzw. $AA. Im Listing 1 war diese Variable auf $20 gesetzt. Die Bedeutung dürfte jetzt klarer sein: Kommandogruppe 0, SCSI-ID 0, Target-Nummer 1 (da der Host-Adapter auf '0' eingestellt war).

## Wieviel ist denn drauf ?

Wenden wir uns einem konkreten Beispiel zu, der Verwendung des Kommandos *READ CAPACITY* (Listing 3). Der Ablauf ist der gleiche wie beim Inquiry-Kommando, nur werden hier 10 Bytes übertragen. Der Opcode ist $25, d.h. Kommandogruppe 1 und Kommando $05. In *Comgroup* ist daher statt $20 der Eintrag $21 zu finden. Nach Aufruf der Funktion SCSI() steht im Feld *Buffer* (u.a.) die Nummer des letzten verfügbaren Blocks. Da die Zählung bei 0 beginnt, errechnet sich die Gesamt-anzahl durch Addition von 1. Ein Block entspricht normalerweise einem Sektor, also 0.5 kB.

Dieses Kommando kann aber noch mehr. Wird im 3. bis 6. Kommandobyte eine Blocknummer eingetragen, und im 9. bzw. vorletzten Byte der Wert $01, so erhält man den letzten Block nach dem vorgegebenen zurück, der noch ohne

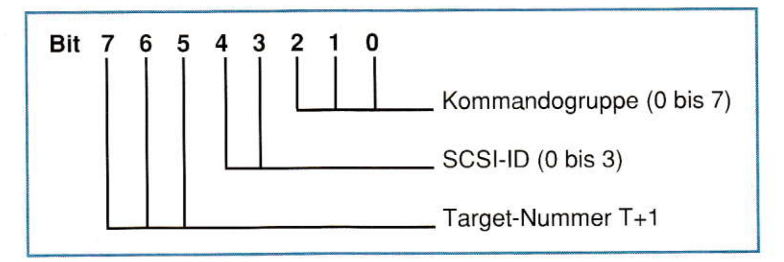



*Tabelle 1*

```
 58:       move.b  d0,dhigh              ; Highbyte
 59:       lea     Comlen,a1             ; Anzahl Kommandobytes
 60:       subq.w  #1,(a1)               ; minus 1
 61:
 62:  loop:
 63:       bsr     setgroup              ; Kommandogruppe einstellen
 64:       move.w  #$88,dmodus           ; ACSI-Bus wachrütteln
 65:       moveq.l #0,d0                 ; d0.l löschen
 66:       clr.w   d1                    ; d1.w initialisieren
 67:  sdloop:
 68:       move.b  0(a0,d1),d0           ; Kommandobyte holen
 69:       bsr     sdbyte                ; und auf Bus legen
 70:       bcs     getstatus             ; Timeout?
 71:       addq.w  #1,d1                 ; d1++
 72:       cmp.w   (a1),d1               ; schon genügend übertragen?
 73:       bne     sdloop                ; nein, weitermachen
 74:
 75:       move.l  #$0a,daccess          ; letztes Kommandobyte, DMA starten
 76:       bsr     waitforcom            ; Übertragung abwarten
 77:       bcs     getstatus             ; Timeout?
 78:
 79:       move.w  #$8a,dmodus           ; Bus selektieren
 80:       nop
 81:       move.w  daccess,d0            ; Status holen
 82:       btst    #1,d0                 ; Fehler?
 83:       bne     getst2                ; jawohl
 84:
 85:       move.l  hz_200,d1             ; Systemzeit holen
 86:       addq.l  #2,d1                 ; 2 Ticks warten
 87:  as_time_goes_by:
 88:       cmp.l   hz_200,d1             ; zwei Ticks weiter?
 89:       bne     as_time_goes_by       ; nein, warten
 90:
 91:       dbf     d2,loop               ; next one
 92:
 93:       move.w  #$8a,dmodus           ; Bus selektieren
 94:
 95:  getstatus:
 96:       move.w  daccess,d0            ; Status holen
 97:  getst2:
 98:       swap    d0                    ; Worthälften tauschen
 99:       move.w  dmodus,d0             ; DMA-Status holen
100:       and.l   #$ff0007,d0           ; ausmaskieren
101:       move.l  d0,status
102:
103:       move.w  #$80,dmodus           ; FDC selektieren
104:       clr     vbl                   ; Floppy-VBL freigeben
105:
106:       move.l  save_ssp,d0           ; Stackpointer zurückholen
107:       bsr     super                 ; in den Usermodus
108:
109:       move.l  status,d0
110:       rts                           ; raus hier
111:
112:  ********************
113:  * super: Zwischen Super- und Usermode umschalten
114:  * IN: D0 neuer Stack
115:  * OUT: D0 alter Stack
116:  * USES: D0, D1, A0, A1 (GEMDOS)
117:  ********************
118:  super:
119:       move.l  d0,-(sp)              ; neue Stackadresse
120:       move.w  #32,-(sp)             ; SUPER
121:       trap    #1                    ; GEMDOS
122:       addq.l  #6,sp                 ; Stack säubern
123:       rts                           ; raus
124:
125:
126:  *********************
127:  * sdbyte: Byte in d0 an HDC schicken
128:  * IN:  d0.l wie folgt:
129:  *        -------------------------
```

→

Verzögerung zu übertragen ist (substantial delay). Üblicherweise ist das die Blocknummer, die vor einem Zylinderwechsel liegt. Damit läßt sich sehr schön die Partitionierung prüfen. Ist die Differenz zwischen übergebener und zurückerhaltener Blocknummer kleiner als 35, brauchen Sie sich nicht über Ihre lahme Platte zu wundern. Dann sind nämlich die wichtigsten Einträge (FAT, Directory) auf zwei Zylinder verteilt. Mit *READ CAPACITY* könnte natürlich gleich das Partitionierprogramm zuverlässig die Zylindergrenzen erkennen.

## Ein Königreich für eine Grafik

Das Bild 2 soll auf anschauliche Weise vermitteln, welchen Weg das erste Kommando-Byte nehmen muß, wenn es ein Gerät erreichen will. Alles beginnt damit, daß aus dem Rechner ein Byte herausschaut. Nehmen wir einmal an, die drei höchstwertigen Bits (TGT = Target-Nummer) lauten binär '100', also vier, dann wird damit der Weg für die restlichen fünf Bits zum ACSI-Gerät 4 durchgeschaltet. Das sei Ihre SH205. Diese Bits werden natürlich als Kommando interpretiert, mit drei Nullen auf 8 Bit erweitert und dem Controller zugeführt. Kein Kommentar.

Der ST-Host-Adapter, das haben Sie sicher schon erraten, ist auf die Adresse 6 eingestellt und belegt damit auf Wunsch auch die Target-Nummer 7. Auf diese Nummer sollen auch die nächsten TGT-Bits eingestellt sein. Die fünf Rest-Bits erhalten nun die Bedeutung ID (2 Bit SCSI-ID) und GRP (3 Bit Kommandogruppe). Sie werden zunächst gespeichert und harren der Dinge, die da kommen. Während des folgenden Zugriffs mit TGT=6 werden die durchgeschalteten Bits wieder als Kommando interpretiert. Allerdings wird das Byte jetzt mit den vorher gespeicherten GRP-Bits aufgefüllt und damit ein vollständiger 8-Bit-Opcode erzeugt. Zusätzlich wählen die ID-Bits einen von vier möglichen Wegen aus.

Das Schema beschreibt nur den Weg des ersten Kommando-Bytes. Alle anderen werden natürlich 1:1 übertragen, also auch das zweite Byte mit der Gerätenummer. Darum kümmert sich der angeschlossene Controller. Die Darstellung der Bussysteme (ACSI/SCSI) als simple Demultiplexer mögen Sie mir verzeihen, auf solche Ideen kommen wohl nur 'Hardwareker'.

```
130:  *       | X.b | X.b | 0.b | CB.b|
131:  *       -------------------------
132:  *       CB=Commandbyte
133:  **********************
134:  sdbyte:
135:        swap    d0                  ; Worthälften vertauschen
136:        move.w  #$8A,d0             ; DMA-Modus einschieben
137:        move.l  d0,daccess          ; Byte auf Bus und Modus an DMA-Chip
138:
139:        moveq   #20,d0              ; 20 Ticks warten
140:  gettimer:
141:        add.l   hz_200,d0           ; d0 plus 200Hz-Zähler
142:  zeita:
143:        btst    #5,gpip             ; auf HDC-IRQ testen
144:        beq     fix_und_fertig      ; ist angekommen, fertig
145:        cmp.l   hz_200,d0           ; Timer-Zielwert erreicht?
146:        bcc     zeita               ; nein, weiter
147:  fix_und_fertig:
148:        rts                         ; und raus
149:
150:  **********************
151:  * waitforcom: Wartet nach Kommandoblock auf Kommandoende
152:  * USES: d0
153:  **********************
154:  waitforcom:
155:        move.l  #800,d0             ; langes Timeout
156:        bra     gettimer            ; und warten
157:
158:  **********************
159:  * setgroup: Kommandogruppe (und SCSI-ID) einstellen
160:  * USES: d0
161:  **********************
162:  setgroup:
163:        move.w  #$88,dmodus         ; Bus wachküssen
164:        move.w  Comgroup,d0         ; Kommandobyte lesen
165:        swap    d0
166:        move.w  #$8a,d0             ; DMA-Modus
167:        move.l  d0,daccess          ; send byte
168:        rts                         ; das war's schon
169:
170:
171:  bss
172:
173:  save_ssp:   ds.l 1              ; Ein Platz für Tiere
174:  status:     ds.l 1              ; Status
175:  Buffer:     ds.b 72             ; DMA-Puffer
176:
177:  end
```

*Listing 1*

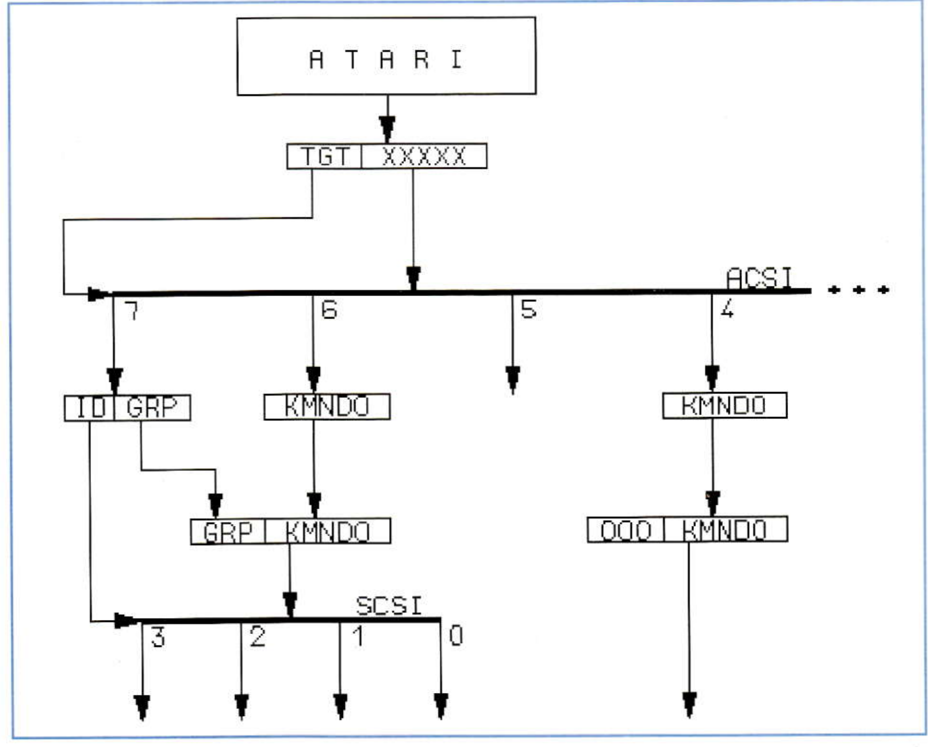



*Bild 2: Auf dem Weg zum Controller*

## Wie geht's weiter ?

Die Verwendung der neuen Möglichkeiten werden neuen oder erweiterten Programmen vorbehalten bleiben. Verschiedene Dienstprogramme werden sinnvoll auf neue Kommandogruppen zugreifen können, zwingend notwendig sind sie sicher nicht. Dem ATARI-Anwender öffnet sich damit aber eine Gerätewelt, die ihm bisher verschlossen war. Als Beispiel möchte ich nur die Streamer anführen, die mit einfachen Gruppe-0-Kommandos gar nicht vernünftig anzusprechen sind. Für den Anschluß mehrerer SCSI-Controller an einen Host-Adapter ist natürlich ein Treiber erforderlich, der die beschriebene Umschaltung beherrscht. Gutinformierte Greise sagen, daß Claus Brod (bekannt aus Funk und Fernsehen) daran arbeitet, seinem CBHD diese Fähigkeit zu vermitteln. Wenn dann noch der gute, alte SED mitmacht...Auch ohne diese Erweiterungen ist der Host-Adapter erwähnenswert, zum Beispiel als Ersatz in der alten SH204. Sollte jemand einen Umbau vornehmen, so möge er uns über das Ergebnis informieren. Ich bin gespannt, welche Art Peripherie dem ATARI in nächster Zeit zugänglich gemacht wird. Geschwindigkeitsprobleme wird es dabei jedenfalls nicht geben. Wenn Sie jetzt immer noch nicht davon überzeugt sind, daß dieses Gerät Ihren Arbeitstisch bereichern sollte, muß ich mein Geld wohl doch wieder mit ehrlicher Arbeit verdienen.

## Zurück in die Hardware Teil 2

Das Wichtigste vorweg: Die bisher existierenden Host-Adapter (ich habe selbst schon 4 Stück gebaut) bewähren sich seit Monaten im Alltagsbetrieb, an der Funktion gibt es also nichts auszusetzen. Trotzdem bemühe ich mich laufend um Verbesserungen, die sich aber auf eine geänderte GAL-Programmierung beschränken sollen. Eine gibt es bereits! Um nach dem Einschalten das IC6 sicherer in seinen Ruhezustand zu bringen, sollten Sie in der Programmieranweisung für das GAL IC5 die Formel für *'SACK'* folgendermaßen erweitern:

```
SACK=(REQ+RST+/SELFF&/A1&/COMP&
/MSG)&ACKFF;
```

Übrigens ist im GAL-Listing 1 der Februarausgabe eine Pin-Definition etwas 'verrutscht'. Vergessen Sie also nicht die Eingabe "D7=4,".

```
 1:  /*********************************************************************/
 2:  /* Beispielprogramm für das SCSI-Kommando INQUIRY           20.01.90 */
 3:  /* Borlands Turbo C, V1.1                            Reiner Wiechert */
 4:  /* (C) MAXON Computer                                                */
 5:  /*********************************************************************/
 6:
 7:  #include <stdio.h>
 8   #include <tos.h>                           /* Megamax: osbind.h      */
 9:
10:  /*********************************************************************/
11:  /* Globale Variablen zur Kommunikation mit dem Assemblermodul        */
12:  /*********************************************************************/
13:
14:  extern  long     SCSI(void);               /* Assemblerfunktion      */
15:  extern  char     Buffer[36];               /* Datenpuffer            */
16:
17:  int     Comgroup = 0x20;                   /* Gruppe 0 / Target 1    */
18:  int     Comlen = 6;                        /* Anzahl Kommandobytes   */
19:
20:  char    Comblock[] = { 0x12, 0, 0, 0, 0x24, 0 };
21:                                             /* Kommandobytes (INQUIRY) */
22:
23:  /*********************************************************************/
24:  /* Hauptprogramm: Anzeige der Gerätekennung (z.b. "SEAGATE ST125N") */
25:  /*********************************************************************/
26:
27:  int main(void)
28:  {
29:       puts("INQUIRY:\n");
30:       printf("Status = %lx\t\t",SCSI());  /* SCSI liefert long-Status */
31:       Buffer[32] = '\n';
32:       Buffer[33] = '\0';                   /* String abschließen      */
33:       puts(&Buffer[8]);
34:
35:       Crawcin();                           /* Auf Tastendruck warten  */
36:       return (0);
37:  }
```

*Listing 2*

```
 1:  /*********************************************************************/
 2:  /* Beispielprogramm für das SCSI-Kommando READ CAPACITY     20.01.90 */
 3:  /* Borlands Turbo C, V1.1    (C) MAXON Computer   by Reiner Wiechert */
 4:  /*********************************************************************/
 5:
 6:  #include <stdio.h>
 7:  #include <tos.h>                           /* Megamax: osbind.h      */
 8:
 9:  /*********************************************************************/
10:  /* Globale Variablen zur Kommunikation mit dem Assemblermodul        */
11:  /*********************************************************************/
12:
13:  extern  long     SCSI(void);               /* Assemblerfunktion      */
14:  extern  char     Buffer[36];               /* Datenpuffer            */
15:
16:  int     Comgroup = 0x21;                   /* Gruppe 1 / Target 1    */
17:  int     Comlen = 10;                       /* Anzahl Kommandobytes   */
18:
19:  char    Comblock[] = { 0x05, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
20:                                             /* Kommandobytes (READ CAP) */
21:
22:  /*********************************************************************/
23:  /* Hauptprogramm: Anzahl der verfügbaren Sektoren ermitteln          */
24:  /*********************************************************************/
25:
26:  int main(void)
27:  {
28:       puts("READ CAPACITY:\n");
29:       printf("Status = %lx\t\t",SCSI());  /* SCSI liefert long-Status */
30:       printf("%ld Sektoren verfügbar\n",(*(long *)Buffer)+1);
31:
32:       Crawcin();                           /* Auf Tastendruck warten  */
33:       return (0);
34:  }
```

*Listing 3*

Wie bereits beschrieben, gab es in der TOS-Version 1.0 ein Problem. Nicht nur das erste, sondern auch das zweite Kommando-Byte wurde mit aktiviertem A1-Signal gesendet. Die aktuelle GAL-Programmierung erkennt darin einen Fehler! Wenn aber unbedingt jemand mit dieser alten Version einen Massenspeicher betreiben möchte - bitte sehr. Zwei geänderte Formeln im GAL IC5 ermöglicht auch das.

1. CSA1=/SELFF&CS&A1&/ACK+MSG&REQ+RST&SELFF;
2. Bei CLACK entfällt innerhalb der Klammer "&/A1".

Den Besitzern von Blitter-TOS (1.2) oder Rainbow-TOS (1.4) kann ich diese Änderung aber nicht empfehlen. Erstens ist sie nicht nötig, und zweitens nimmt sie dem Host-Adapter etwas von seiner Sicherheit (Knautschzone) gegenüber herumstreunenden DMA-Programmen. Der Unterschied liegt darin, ob nach einem fehlerhaften Zugriff auf den DMA-Chip der Bus belegt bleibt oder nicht. Solche Fälle dürften aber höchstens dann auftreten, wenn Sie mit selbstgeschriebenen Routinen experimentieren.

*Noch ein Tip:* Verwenden Sie die Target-Adressen 2, 4 oder 6, wenn es möglich ist. Wird der ATARI eingeschaltet, wackeln die Signale auf dem ACSI-Bus dermaßen, daß eine Kombination von aktiven CS- und ACK-Signalen möglich ist. Damit kann dann zufällig Target-Nummer 0 angesprochen werden. Die Verwendung einer anderen Nummer ist der einfachste Weg, diesem Problem aus dem Wege zu gehen. Damit sollten aber auch die restlichen Klarheiten beseitigt sein. Ein überschwengliches Schlußwort werde ich Ihnen und mir ersparen. Bis dann.

*Reiner Wiechert*

```
 1:  ****************************************
 2:   Dokumentation: BOOLEAN-EQUATIONS
 3:   Datei: SCS05V24.PLD
 4:   Datum: 12.01.90      Uhrzeit: 18:30
 5:  ****************************************
 6:
 7:  SCSI-INTERFACE
 8:  IC5,     GAL FÜR STEUERUNGSAUFGABEN
 9:  FAST-MODUS
10:
11:
12:  *IDENTIFICATION
13:   SCS05V24;
14:
15:  *TYPE
16:   GAL20V8;
17:
18:  *PINS
19:   /DRQ    =  2,        % DRQ eines folgenden ACSI-Gerätes          %
20:   /ACK    =  3,        % ACSI Acknowledge                          %
21:   /A1     =  4,        % ACSI Anzeige für 1. Kommandobyte          %
22:   /SELFF  =  5,        % Merker: 'Hostadapter selektiert' aus IC 2 %
23:   /CS     =  6,        % ACSI Chip-Select                          %
24:   /ACKFF  =  7,        % Gespeichertes CS/ACK für Handshaking
25:   /IO     =  8,        % SCSI Input/Output                         %
26:   /REQ    =  9,        % SCSI Request
27:   /MSG    = 10,        % SCSI Message                              %
28:   /BSY    = 11,        % SCSI Busy                                 %
29:   COMP    = 14,        % Vergleichssignal as IC 2                  %
30:   /RST    = 23,        % Reset                                     %
31:
32:   /CSA1.T   = 15,      % Takt für IC 2                %
33:   CLRD.T    = 16,      % Takt für Leseflipflop        %
34:   /ENWR.T   = 17,      % Freigabe für Schreiblatch    %
35:   AREQ.T    = 18,      % ACSI-Request -> INT/DRQ      %
36:   ENBUF.T   = 19,      % Freigabe für DMA-Puffer      %
37:   /SACK.T   = 20,      % SCSI Acknowledge             %
38:   SELOUT.T  = 21,      % Signal zur SCSI-Selektion    %
39:   CLACK.T   = 22;      % Takt für Handshake-Flipflop  %
40:
41:  *BOOLEAN-EQUATIONS
42:
43:    CSA1    = CS & A1 & /ACK  +  MSG & REQ  +  RST & SELFF;
44:
45:    SELOUT = SELFF & /BSY;
46:
47:    CLRD    = REQ & IO & SELFF  +  /SELFF & ENBUF & (CS + CLRD) + MSG;
48:
49:    ENWR    = BSY & /IO;
50:
51:    AREQ    = SELFF & /ACKFF & REQ  +  /SELFF & DRQ;
52:
53:    ENBUF   = SELFF  +  DRQ  +  ENBUF & (CS & /ACK  +  /CLRD & /RST);
54:
55:    SACK    = (REQ + RST + /SELFF & /A1 & /COMP & /MSG) & ACKFF;
56:
57:    CLACK   = MSG & REQ  +  /RST & SELFF & (ACK + CS & /A1)
58:            + CS & A1 & COMP & /ACK;
59:
60:  *END
```

*Listing 4*


***Literatur:***

*Brod/Stepper: Scheibenkleister II*

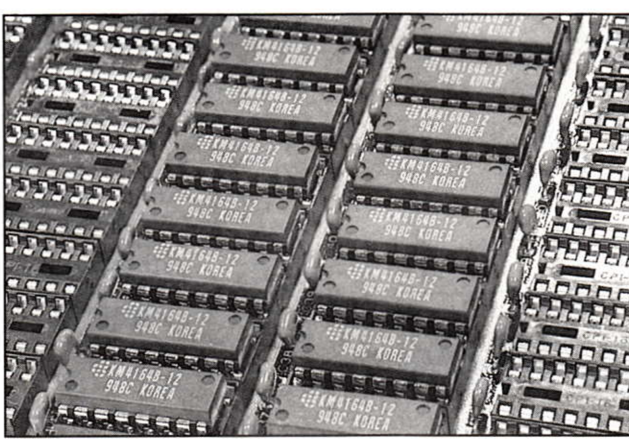
*SEAGATE Technology: ST125N/138N/157N Product Manual, Rev.C*

*Lattice: GAL Handbook*

# Aus zweieinhalb mach drei

## Optimale Ausnutzung von Speichererweiterungen

**Dieser Artikel wendet sich an alle ST-Besitzer mit 1 MB Hauptspeicher (1040, Mega 1, aufgerüstete 520 oder 260), die ihren Speicher um 2 MB erweitert haben. Nun sind Sie stolzer Besitzer von 2,5 MB Hauptspeicher, trauern aber den 512 kB nach, die ungenutzt im Rechner liegen.**



Die hier vorgestellte Schaltung aktiviert diese 512 kB wieder, so daß Sie dann 3 MB vollwertigen Hauptspeicher zur Verfügung haben. Und das alles ohne zusätzliche Software. Die Kosten für die ICs belaufen sich ca. auf 2,50 DM, dazu kommen 4 Widerstände à 68Ω, ein Blockkondensator von 1 µF und ein kleines Stück Lochrasterplatine. Bastel- und Löterfahrung sollten aber vorhanden sein.

Außerdem wird eine Speichererweiterung um 2 MB hierdurch auch kostenmäßig interessant, auch für Leute, die noch vorhaben, ihren Speicher zu erweitern. Man erweitert den Speicher netto um 2 MB, und nicht nur um 1,5 MB. Mein 1040 ST läuft seit dem 1.1.1990 mit 3 MB, und es hat bis jetzt keine Probleme gegeben. Ich habe die Schaltung mit RAM-TOS 1.0, ROM-TOS 1.0 und ROM-TOS 1.4 getestet. Ein 24stündiger Speichertest ergab keine Fehler.

Bevor ich zu der eigentlichen Schaltung komme, etwas Theorie vorweg: Der Hauptspeicher des ST besteht aus dynamischen RAMs (DRAM). Für jede Speicherstelle wird nur ein einziger Transistor verwendet. Dies ermöglicht eine hohe Geschwindigkeit, eine geringe Verlustleistung und eine hohe Packungsdichte. Die Speicherzellen benötigen aber eine ständige Auffrischung (Refresh), um die Daten nicht zu verlieren. Die Refresh-Zeit jeder Speicherzelle liegt ca. bei 2 ms. Gegenüber statischen RAMs ist der externe Schaltungsaufwand bei DRAMs aber erheblich größer. Diese zusätzlichen Aufgaben übernimmt im ST die MMU (Memory Manager Unit - Speicherverwaltungseinheit).

Die DRAMs (jedenfalls die, von denen hier die Rede ist) sind bitweise orientiert, d.h. jedes IC hat nur eine Datenleitung. Man braucht also für einen 16 Bit-Datenbus wie beim ST auch immer 16 Speicher-ICs. Diese 16 ICs nennt man auch eine RAM-Bank. Die Adreßsignale der DRAMs sind gemultiplext, d.h. auf einer Adreßleitung des DRAM liegen zwei Adressen, die zeitlich nacheinander übertragen werden. Im DRAM werden diese Adressen wieder getrennt. Damit hält man die Anzahl der Anschluß-Pins so gering wie möglich. Das Multiplexen der Adressen und das Erzeugen der Steuersignale übernimmt die MMU. Der gemultiplexte Adreßbus der DRAMs ist auch der Grund dafür, warum die DRAMs nur in Viererschritten größer werden. Es gibt z.B. keine 512 kBit- oder 2 MBit-DRAMs, sondern nur die Typen 16 kBit, 64 kBit, 256 kBit, 1 MBit und 4 MBit, wobei im Moment der 4 MBit-Baustein noch nicht in Massenstückzahlen produziert wird.

Für meine Schaltung interessant sind die beiden Steuerleitungen RAS und CAS. Beide Signale sind low-aktiv. Mit RAS wird die niederwertige Adresse der gemultiplexten Adreßleitungen, mit CAS die höherwertige Adresse übernommen (siehe auch Impulsdiagramm). Außerdem wirken RAS und CAS gleichzeitig aktiv wie Chip-Select und geben den Speicherbaustein frei. Mit dem Signal WE wird einSpeicherschreib-oder Lesezugriff unterschieden.

Beim ATARI ST ist die CAS-Leitung noch aufgeteilt in CASL und CASH. CASL geht an die Speicher-ICs mit den Datenleitungen 0-7 und CASH an die mit den Datenleitungen 8-15. Hiermit kann man den Speicher auch byteweise beschreiben, entweder das Low-Byte oder das High-Byte. Bei einem Wortzugriff werden CASL und CASH gleichzeitig aktiv.

Die MMU im ST kann von sich aus zwei RAM-Bänke verwalten. Für die Bank 0 dienen die Steuerleitungen RAS0, CAS0L und CAS0H, für die Bank 1 die Leitungen RAS1, CAS1L und CAS1H. Die Bänke können mit ICs der Typen 64 kBit, 256 kBit und 1 MBit bestückt werden. Den 64 kBit-Baustein wollen wir mal schnell vergessen, er wurde und wird wohl auch nicht mehr verwendet. Man kann damit auch nur 128 kByte in einer RAM-Bank unterbringen.

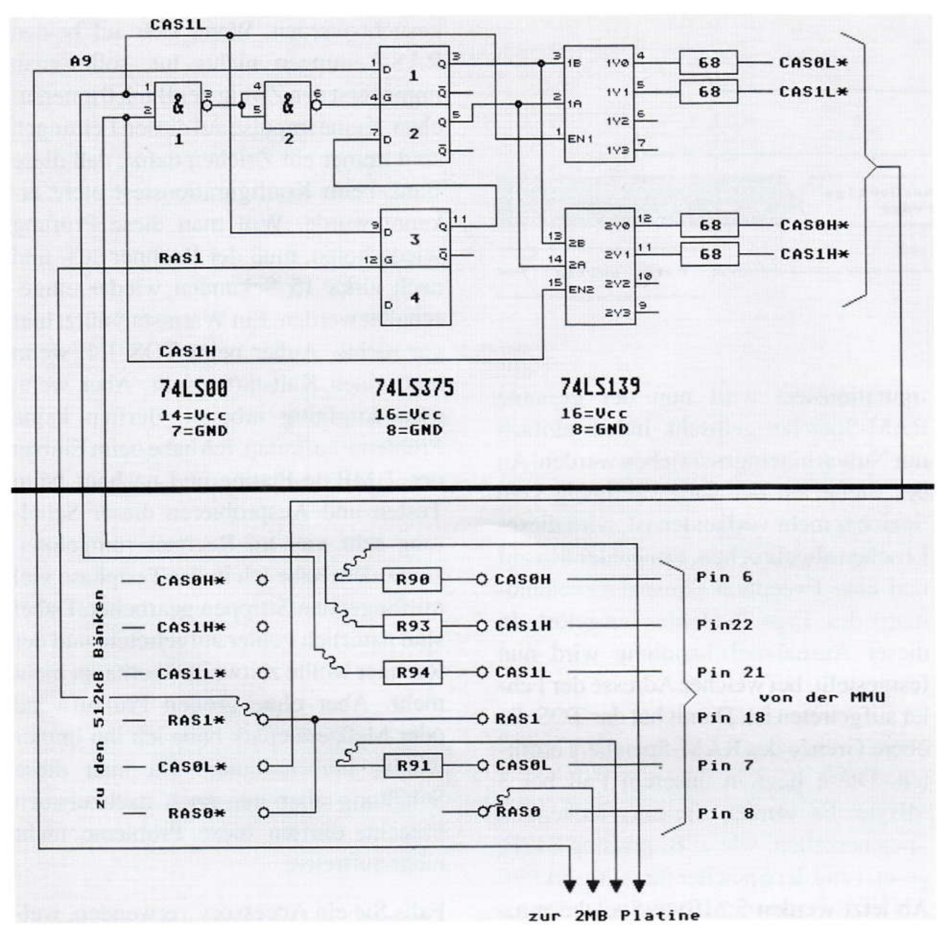



*Bild 1: Das Schaltbild*

Es gibt mehrere Möglichkeiten der Bestückung mit DRAMs, wobei in der RAM-Bank 0 immer Speicher vorhanden sein muß. Innerhalb einer Bank dürfen natürlich nur ICs gleicher Größe verwendet werden.

1. eine Bank mit 256 kBit ICs = 512 kB
2. beide Bänke mit 256 kBit ICs = 1 MB
3. eine Bank mit 1 MBit ICs = 2 MB
4. beide Bänke mit 1 MBit ICs = 4 MB
5. eine Bank mit 256 kBit ICs, eine Bank mit 1 MBit ICs = 2.5MB

Die Konfiguration der RAM-Bänke wird beim Kaltstart (nur beim Kaltstart, nicht beim Warmstart) des Rechners vom TOS getestet und das Ergebnis in dasKonfigurationsregister (Adr.$FF8001) der MMU eingetragen. Außerdem findet man dieses Konfigurations-Byte auch noch im RAM an der Adresse $424 (memctrl). Dieses Byte kann folgende Werte haben:

1.= $04, 2.= $05, 3.= $08, 4.= $0A, 5.= $06, oder $09.

Diese Werte beziehen sich direkt auf die oben genannten Bestückungsmöglichkeiten mit DRAMs.

Jede Bank hat einen Adreßbereich von 2 MB. Sind in Bank 0 aber nur 512 kB vorhanden, setzt die MMU automatisch den Adreßbereich von Bank 1 an den von Bank 0 heran. Dadurch ensteht ein durchgehend adressierbarer Speicherbereich, den das TOS auch erwartet. Ansonsten würde das TOS zum Bombenleger.

Und nun zur Erklärung der Schaltung: Wie wir bis jetzt gehört haben, kann die MMU nur zwei Speicherbänke verwalten, wir wollen aber drei Bänke (möglich sind mit der Schaltung sogar fünf) anschließen.

Dazu brauchen wir von der MMU die Adreßleitung A9, die an den 256 Kbit ICs nicht vorhanden ist. Die Adreßleitung A9 ist gemultiplext und beinhaltet die realen Adressen A18 und A19. Die beiden NAND-Gatter dienen als ODER-Gatter und sollen das D-Latch 1 und 2 freigeben, wenn CAS1L oder CAS1H aktiv werden. Bei einem Byte-Zugriff auf den Speicher wird nur CASL oder CASH aktiv, je nachdem, ob der Zugriff auf eine gerade oder ungerade Adresse erfolgt. Das IC 74LS375 dient als Demultiplexer und holt aus der Adreßleitung A9 die Adressen A18 und A19. Mit dem D-Latch 3 wird die Adresse A18 und mit dem D-Latch 1 die Adresse A19 erzeugt. A18 wird noch über das D-Latch 2 geführt. Hiermit wird A18 solange gehalten, bis CAS auf high geht, und nicht schon ungültig wird, wenn RAS auf high geht (siehe auch Impulsdiagramm). Das IC 74LS139 (zwei 2-Bit Binärdecoder) erzeugt mit den Adressen A18 und A19 und den Signalen CAS1L und CAS1H acht neue CAS-Signale, von denen wir allerdings nur vier brauchen. Diese vier CAS-Signale werden den beiden vorhandenen 512 kByte-RAM-Bänken zugeführt. Wie man in der Schaltung sieht, sind die Pins 1Y2, 1Y3, 2Y2 und 2Y3 des ICs 74LS139 noch frei. Hier könnte man sogar noch zwei weitere 512 kByte-RAM-Bänke anschließen und hätte somit 4 MByte Hauptspeicher zur Verfügung. Es wären dann sogar fünf RAM-Bänke vorhanden. Man könnte mit dieser Schaltung theoretisch auch 1 MB-STs mit zwei zusätzlichen 512 kByte RAM-Bänke auf 2 MB erweitern. Dazu müßte man diese vier 512 kByte-RAM-Bänke nur auf die Bank 0 der MMU legen. Ob sich dies allerdings finanziell lohnt, kann ich nicht sagen. Falls jemand aber 256-kBit-Bausteine günstig bekommen kann, wäre es zu überlegen. Außerdem ist der Platzbedarf für zwei weitere RAM-Bänke zu bedenken, und der ist im 1040 ST nicht gerade üppig vorhanden.

Ich habe diese Schaltung auf einer Lochrasterplatine in Fädeltechnik aufgebaut. Der Aufwand ist sehr gering und sollte für jeden Bastler zu schaffen sein. Was sehr wichtig ist: die 2 MByte-Platine muß als Bank 0 angeschlossen werden. Die RAS0-Leitung mußte ich bei meiner Hauptplatine von unten durchtrennen, es soll aber auch Platinen geben, auf denen sich ein 0 Ω Widerstand befindet. Dieser muß dann ausgelötet werden. Es ist auch sehr wichtig, daß die Zuordnung der RAS-und CAS-Leitungen stimmt. Am besten mit einem Ohmmeter von der MMU her durchprüfen. Es soll 12 verschiedene Hauptplatinen des ST geben, und die Widerstandsbezeichnungen müssen nicht mit denen meiner Platine übereinstimmen. Mein 1040 ST ist übrigens Baujahr Juli 1987.

Hier eine Übersicht über die wichtigen Pins der MMU:

6=CAS0H, 7=CAS0L, 8=RAS0, 18=RAS1, 21=CAS1L, 22=CAS1H, 64=A9

Die Widerstände R90, R93, R94 und R91 habe ich auf der Seite, die zu den beiden original 512 kByte-RAM-Bänken geht,

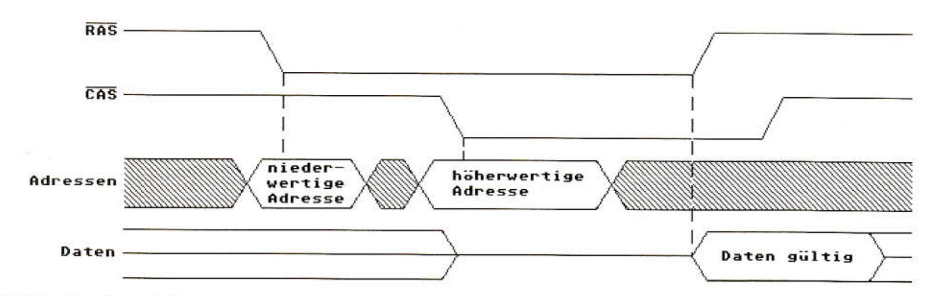



*Bild 2: Das Impulsdiagramm*

hochgelötet. An die hochgelöteten Seiten der Widerstände habe ich die Leitungen von der 2 MByte-Platine und meiner Schaltung gelötet. An die Punkte (in meiner Schaltung mit CAS0H* usw. bezeichnet), die zu den beiden 512 kByte-Bänken gehen, habe ich die vier Leitungen von dem IC 74LS139 angelötet. Die RAS1-Leitung wird einfach parallel zur RAS0-Leitung (die aufgetrennt sein muß) gelötet.

Die Adreßleitung A9 nimmt man am besten von der 2 MByte-Platine ab, und zwar hinter dem 33 Ω Widerstand. Ansonsten kann man A9 auch direkt von der MMU abnehmen. Beim 1040 ST ist A9 (auch MAD 9 genannt) weiter nicht beschaltet. Es sollte dann nur ein 33 Ω Widerstand in die Leitung gelötet werden. Auf Leitungslängen und Verlegung der Leitungen im Rechner muß nicht besonders geachtet werden. Zu lang sollten die Leitungen allerdings auch nicht sein. Plus und minus 5V habe ich an dem großen Kondensator vor dem Stecker vom Netzteil abgenommen.

Hier noch ein Tip für Leute, die sich eine 2 MByte-Platine selbst bauen oder einbauen wollen. Achten Sie auf eine genügend dimensionierte Masseleitung auf der Platine. Ich hatte bei mir zu Anfang dauerndes Bildschirmflimmern, und das Speichertestprogramm zeigte viele Fehler an. Nachdem ich die Masseleitung auf der 2 MByte-Platine verstärkt hatte, war der Spuk vorbei.

Nun noch die Erklärung, warum keine zusätzliche Software erforderlich ist. Wie schon vorher erklärt, testet das TOS nach einem Kaltstart, welche RAM-Bänke vorhanden und wie groß diese sind. In unserem Fall stellt das TOS 2 mal 2 MByte RAM-Bänke fest. An der Speicherstelle memctrl $424 muß also $0A stehen. Jetzt ist aber die RAM-Bank 1 nicht 2 MByte groß, sondern nur 1 MByte. Hier kommt uns das TOS wieder zu Hilfe. Nach diesem Speicherkonfigurationstest wird nun der gesamte RAM-Speicher gelöscht, indem einfach nur Nullen hineingeschrieben werden. An der Stelle, an der hardwaremäßig kein Speicher mehr vorhanden ist, wird dieses Löschen abgebrochen. Ein Fehler tritt auf und eine Exception (Ausnahmebehandlung) des Typs Busfehler ausgelöst. In dieser Ausnahmebehandlung wird nun festgestellt, bei welcher Adresse der Fehler aufgetreten ist. Damit hat das TOS die obere Grenze des RAM-Speichers ermittelt. Diese liegt in unserem Fall bei 3 MByte. Es werden die entsprechenden Speicherzellen, wie z. B. phystop $42E, gesetzt und der Speicher für gültig erklärt. Ab jetzt werden 3 MByte-Speicher problemlos verwaltet.

Dies ist auch der Grund, warum die 2 MByte-Platine RAM-Bank 0 sein muß. Die Speicherkonfiguration geht von 4 MByte Speicher aus. Wenn am Ende 1 MByte fehlt, macht das nichts aus. Fehlte aber in der Mitte des Speichers 1 MByte, würde das Löschen des Speichers schon nach einem MByte abgebrochen. Man hätte dann 1 MByte-Hauptspeicher und 2 MByte Speicher würden tot herumliegen. Und das wollen wir doch erst recht nicht, oder?

Die wichtigsten Punkte noch einmal in Kurzform:

1. auf sauberes Löten achten
2. Die 2 MByte-Platine muß Bank 0 sein.
3. die Anschlüsse RAS und CAS, wenn nötig von der MMU her mit dem Ohmmeter ausprüfen.

Falls der Rechner nach dem Einbau dieser Schaltung nicht laufen sollte, würde ich erst einmal alle Lötstellen überprüfen. Falls dann immer noch nichts geht (was ich nicht hoffe), hier noch ein Tip. Mit einem Logiktester oder Oszilloskop die beiden Leitungen RAS1 und RAS0 beim Einschalten des Rechners beobachten. Zeigen sich auf einer dieser Leitungen keine Impulse, muß in diesem Zweig gesucht werden. Wenn sich auf beiden RAS-Leitungen nichts tut, sollte man immer erst den Zweig der Bank 0 untersuchen. Keine Impulse auf diesen Leitungen sind immer ein Zeichen dafür, daß diese Bank beim Konfigurationstest nicht erkannt wurde. Will man diese Prüfung wiederholen, muß der Rechner aus- und nach zirka 15 Sekunden wieder eingeschaltet werden. Ein Warmstart nützt hier gar nichts. Außer beim TOS 1.4, wenn man einen Kaltstart macht. Aber wenn man sorgfältig arbeitet, dürften keine Probleme auftreten. Ich habe beim Einbau der 2 MByte-Platine und nachher beim Testen und Ausprobieren dieser Schaltung sehr viel im Rechner rumgelötet. Außerdem habe ich in der Testphase viel mitfliegenden Strippen gearbeitet. Dabei sind natürlich Fehler aufgetreten, und der Rechner wollte zeitweise überhaupt nicht mehr. Aber ohne großen Prüfaufwand oder Meßgerätepark habe ich ihn immer wieder hinbekommen. Da man diese Schaltung aber nur noch nachzubauen braucht, dürften diese Probleme nicht mehr auftreten.

Falls Sie ein Accessory verwenden, welches den freien Speicherplatz anzeigt, wundern Sie sich bitte nicht, wenn ein freier Speicherplatz unter 3 Millionen Bytes angezeigt wird. Erstens braucht das TOS selbst einen gewissen Speicherplatz (abhängig von der TOS-Version), und zweitens brauchen alle Accessories, die geladen sind, Speicherplatz. Da dieser nicht mehr frei ist, wird er auch nicht angezeigt. Nebenbei gesagt, verbrauchen die meisten Programme (auch Accessories) mehr Speicherplatz im RAM, als sie auf der Diskette oder Harddisk groß sind. Ein Programm, welches auf der Diskette 10 kByte groß ist, kann im RAM gut und gerne 11, 100, 200 kByte oder auch mehr belegen. Wenn man ganz sichergehen will, ob man auch wirklich 3 MByte Speicher hat, sollte man sich mit einem Speichermonitor die Speicherstelle *phystop* an derAdresse $42E ansehen. Hier muß der Wert $300000 stehen. Zeigt das RAM-Frei Programm allerdings einen Wert über 2,6 Millionen Bytes an, kann man davon ausgehen, daß 3 MByte vorhanden sind. Ich möchte noch kurz die in diesem Artikel verwendeten Begriffe Kaltstart und Warmstart erklären, da ich davon ausgehe, daß nicht jeder damit etwas anfangen kann. Ein Warmstart entspricht dem Druck auf die Reset-Taste, ein Kaltstart wird beim Einschalten des Rechners gemacht. Nach dem Einschalten stehen im RAM zufällige Werte,

wobei das Wort "zufällig" nicht so wörtlich zu nehmen ist. Bei meinem alten 8-Bit-Rechner mit 8085-Prozessor standen nach dem Einschalten immer 256 Byte $FF und 256 Byte $00 abwechselnd im DRAM. Ich nehme an, im ST wird es genauso oder ähnlich sein. In der Intialisierungsphase des TOS' werden bestimmte Speicherstellen abgefragt. Stehen hier bestimmte Werte, wird der Speicher für gültig erklärt und Konfigurations- und Speichertest und einiges mehr nicht durchgeführt. Resetfeste Programme (auch ein Virus) bleiben im Speicher. Dies entspricht dem Warmstart. Bei einem Kaltstart steht an diesen Stellen irgendwas, nur nicht das, was sollte. Es werden die gesamten Initialisierungsroutinen durchlaufen. Dabei werden natürlich auch resetfeste Programme, falls vorhanden, rausgeworfen (das mit den resetfesten Programmen gilt natürlich nur für einen Software-Kaltstart, nach dem Einschalten des Rechners befinden sich keine Programme im RAM). Will man also ganz sichergehen und hat keine resetfeste RAM-Disk mit noch nicht abgespeicherten Daten im Speicher, sollte man immer einen Kaltstart machen. Man kann einen Warm-oder Kaltstart auch softwaremäßig auslösen. Ein Beispiel für einen Software-Kaltstart finden Sie weiter unten. Ab TOS 1.4 kann man den Warm- sowie Kaltstart über die Tastatur auslösen.

Zum Schluß noch viel Spaß beim Bauen und Einbauen dieser Schaltung und viel Spaß mit 3 MByte.

*K.-D. Litteck*

Bauteileliste:

- 1 kleines Stück Lochrasterplatine
- 1 IC 74LS00
- 1 IC 74LS375
- 1 IC 74LS139
- 4 Widerstände à 68 Ω
- 1 Blockkondensator
- 1 µF Fädeldraht und/oder Kupferlitze




# ADIMENS ST und DAILY MAIL

**Die vielfältigen Möglichkeiten von Adimens bezüglich der Übernahme von Daten fremder Programme sind hinlänglich bekannt. Exemplarisch ist der Austausch von Daten mit dem Programm Daily Mail. Dieses Programm beinhaltet neben einem Texteditor eine kleine Dateiverwaltung.**

Die Maske dieser Dateiverwaltung ist fest vorgegeben und kann vom Benutzer weder hinsichtlich weiterer Merkmale noch hinsichtlich der Feldlängen der vorhandenen Merkmalsfelder verändert werden.

Wer die mit Daily Mail gesammelten Adressen in eine professionelle Datenbank wie Adimens einbinden will, hat es nicht schwer. Das Datenformat beider Programme ist im Klartext ex- bzw. importierbar.

Die Maske von Daily Mail ist wie folgt aufgebaut (Merkmalsname, Feldlänge): *Anrede (20), Vorname (20), Name (30), Ergänzung (30), Straße (30), Plz - Ort (30), Telefon (15), Suchbegriffe (30), Anredeformel (35).* Beim Export der Daten hat man die Wahl zwischen drei Exportformaten (s.u.). Die Daten werden nicht in der Reihenfolge der Maske ausgegeben, sondern mit einer kleinen Änderung: Als erstes wird der Name ausgegeben, dann der Vorname, dann die Anrede, danach geht es allerdings weiter wie in der Maske. Warum der Programmierer die Reihenfolge beim Export anders festgelegt hat als in der Maske, bleibt zwar ein Rätsel, ist aber nicht weiter tragisch. In Adimens kann man die Maske so gestalten, daß die Reihenfolge der Merkmalsfelder mit der Exportfolge von Daily Mail übereinstimmt, kann aber auch die Felder frei plazieren und mittels der Option "Reihenfolge ändern" den Import steuern.

Schade ist nur, daß Felder in Daily Mail, bei denen die vorgegebene Feldlänge nicht vollständig ausgenutzt wurde, mit Leerzeichen aufgefüllt wurden. In Adimens werden dann diese Leerzeichen mit importiert. Das Problem ist aber lösbar über AdiTALK. Mit dem dort enthaltenen TRIM-Befehl lassen sich die überflüssigen Leerzeichen löschen.

Daily Mail bietet drei verschiedene Exportformate an, die dort Format 1, Format 2 und Format 3 heißen. Gleich welches man wählt, lassen sich die Daten in Adimens importieren.

Import-/Export-Parameter Adimens bei Export Daily Mail Format 1:

Datensatztrennung : (13)(10)
Feldtrennung : ,
Feldeinfassungen : "
Dezimalpunkt: .
Datum : TT/MM/JJ

Import-/Export-Parameter Adimens bei Export Daily Mail Format 2:

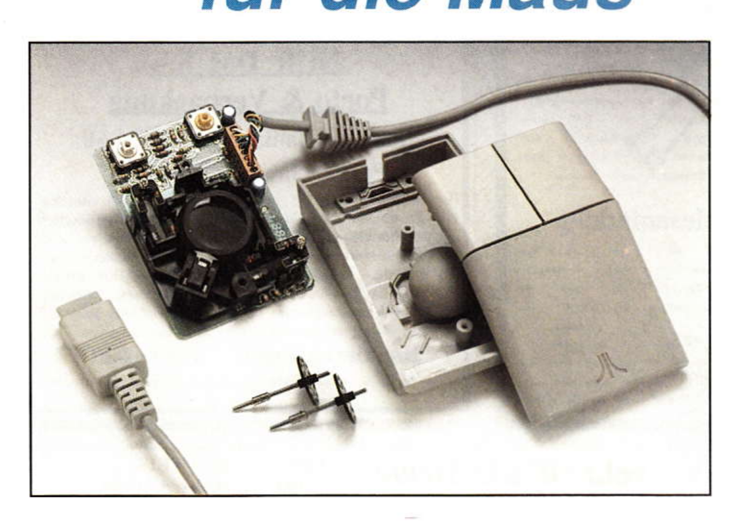
Datensatztrennung : (13)(10)
Feld-Trennung : ,
Feldeinfassungen : Dezimalpunkt: .
Datum : TT/MM/JJ

Import-/Export-Parameter Adimens bei Export Daily Mail Format 3:

Datensatztrennung : (13)(10)
Feldtrennung : (13)(10)
Feldeinfassungen : Dezimalpunkt: .
Datum : TT/MM/JJ

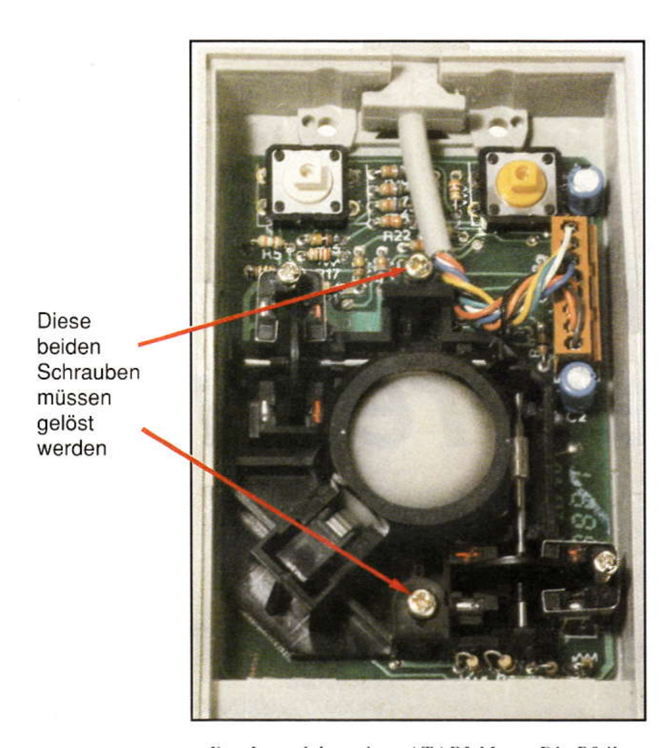
Viel Erfolg bei der Übernahme der Daten nach Adimens!

# Erste Hilfe für die Maus



**Wer kennt ihn nicht, den ausgeleierten Mausknopf, den man nur nach vielen Versuchen (und unter Androhung von Gewalt) zum Doppelklick bewegen kann? Welcher ST-Benutzer ist noch nicht verzweifelt, wenn der Pfeil nicht so will, wie man die Maus bewegte? Wer hat nicht schon mit einem Schraubenzieher oder einem Wattebausch in der unteren Öffnung der Maus die Walzen gereinigt, damit sein Tier ihm wieder brav folgt?**

Wenn man sich die Unterseite der Maus einmal näher ansieht, entdeckt man zwei Schrauben, die nach dem Öffnen das Innenleben der Maus freigeben (Vorher sollte man den 'Mausball' durch Öffnen der Mausklappe entfernen). Nun sind die Übeltäter frei zugänglich. Im oberen Teil sind die beiden Schalter für die Maustasten zu sehen. Über und rechts von der Führung für den 'Mausball' sind die (vermutlich) mit Staubfäden umwickelten Walzen zu entdecken. Zunächst muß die Platine gelöst werden. Dazu müssen wieder zwei Schrauben (siehe Pfeile) geöffnet werden. Die anderen beiden Schrauben sollte man tunlichst in Ruhe lassen. Sie dienen zur exakten Fixierung der




*Das Innenleben einer ATARI-Maus. Die Pfeile bezeichnen die Schrauben, die zum Herausnehmen der Platine herausgedreht werden müssen.*

Abtastdioden und sind mit Lack verklebt. Hat man die Platine nun aus der Gehäuseunterschale entnommen, können die Walzen zur besseren Reinigung aus ihren Führungen herausgeholt werden. Dazu jeweils den äußeren Hebel leicht nach außen beugen und die Walzen entnehmen. Sind sie wieder in glänzender und staubfreier Verfassung, können sie durch einfaches Drücken wieder in die Halterungen zurückgebracht werden, dies sollte man jedoch erst dann machen, wenn der Mausknopf gewechselt worden ist.

Zum Wechseln des [oder der (aber in den meisten Fällen dürfte wohl nur der linke Schalter nicht mehr so richtig wollen)] Schalters, muß er (oder sie) erst ausgelötet werden. Am besten hilft man sich dabei mit einer Entlötsaugpumpe oder Entlötlitze. Ist der defekte Schalter entfernt und der neue auf seinem Platz, kann die Maus wieder zusammengebaut werden. Es ist übrigens egal, wieherum der Schalter eingebaut wird, solange die Führungsstifte in die entsprechenden Löcher gelangen.

Bleibt zum Schluß eigentlich nur noch eine Frage zu klären: Wo bekommt man die Schalter? Die in der ATARI-Maus verwendeten Schalter sind von OMRON. Nun gibt es verschiedene Typen, die sich einmal in der Betätigungskraft und zum anderen in der Lebensdauer unterscheiden. Der Standardtyp ist B3F-4050.

Wer es gerne doppelt so schwer haben möchte, seine Maustaster zum Klicken zu bringen, dem sei der Typ B3F-4055 empfohlen. Und all' jene, denen eine Haltbarkeit von 3*106 Schaltspielen (das sind 3 Millionen Klicks!!!) nicht ausreicht, können den Schalter auch mit Silber- (B3F-5050) oder Goldkontakten (B3F-5051) bekommen.

***Der Preis?***

DM 1,- (eine Deutsche Mark) pro Stück (Standdardtyp).

Natürlich kommen die üblichen Versandkosten noch hinzu, aber mit ein bißchen Bastelarbeit spart man viel Geld. Und nun viel Spaß und gutes Klikken mit Ihrer (fast) neuen Maus.

*TW*

***Bezugsadresse:***

*CANNING CED*
*Ing. Theo Henkes GmbH*
*Postfach 721226*
*3000 Hannover 72*




# Haben auch Sie einen *Quick-Tip*?

**Standen Sie auch einmal vor einem kleinen, aber schier unlösbarem Problem? Dann, durch Zufall bekamen Sie einen Tip und schon war es gelöst.**

**Ähnlich haben wir uns diese neue Rubrik in der ST Computer vorgestellt. Aufgerufen sind auch Sie, liebe Leser(innen)! Geben Sie Ihre Erfahrungen weiter, egal, ob es um Anwendungen, Programmieren o.ä. geht.**

**Wir sammeln Ihre (und unsere) Tips und stellen Sie ggf. in den Quick-Tips vor.**

**Beteiligen Sie sich an der ST Computer!**

Einsendungen an: MAXON Computer
ST Computer Redaktion
Stichwort: Quick-Tip
Industriestr. 26
6236 Eschborn

# Desktop-Spielereien

**"Warum kann ich den Papierkorb nicht öffnen? Warum kann ich keinen Mülleimer in ein Fenster legen? Wieso lassen sich die Ordner nicht auf das Desktop verfrachten? Wieso ist der Desktop-Hintergrund immer grün? Warum druckt mein Drucker immer falsch? Warum ist der Tastenklick immer an? Warum ist die Schnittstelle immer falsch eingestellt?" - Diese Fragen werden Sie sich bestimmt auch schon gestellt haben. Frei nach dem Motto "mit viel Rubbeln ist jetzt Schluß" können Sie Ihr Desktop auch so ändern, daß es immer nach Ihren Vorstellungen eingestellt ist!**

Um nicht nach jedem neuen Booten das Desktop den eigenen Wünschen anpassen zu müssen, kann man den aktuellen Stand durch *Arbeit sichern* abspeichern. Dabei wird eine Datei namens *DESKTOP.INF* angelegt, die Informationen über RS232- und Druckerkonfiguration, Kontrollfeldparameter, Bildschirmauflösung, Position der Fenster und Icons und Anwendungen enthält. Haben Sie schon einmal versucht, diese Datei in einen Texteditor einzuladen? Wenn nicht, sollten Sie es unbedingt einmal probieren - viele erstaunliche Dinge können geändert, ergänzt oder gelöscht werden!

Alle Informationen werden durch ein Doppelkreuz und einem nachfolgenden Buchstaben zur Kennung eingeleitet. Hinter dieser Kennung steht dann die Parameterliste. Sie besteht entweder aus einzelnen Ziffern, deren Bedeutung in den nachfolgenden Tabellen erklärt wird, oder aus Hex-Zahlen in Ziffernschreibweise. So kann H1 (vom Kontrollfeld) eine Zahl zwischen 00 (0) und FF (255) sein.

## RS232-Einstellung

#a123456

| | |
|---|---|
| 1. Duplex | 0=Voll, 1=Halb |
| 2. Baud | 0=9600, 1=4800, 2=1200,3=300 |
| 3. Parität | 0=Keine, 1=Odd, 2=Even |
| 4. Datenbits | 0=8, 1=7, 2=6, 3=5 |
| 5. RTS/CTS, XON/-OFF | 0=Aus/Aus, 1=Aus/An, 2=An/Aus, 3=An/An |
| 6. High Bit | 0=An, 1=Aus |

Die Desktop-Info in Bild 1 hat also die Einstellungen Vollduplex, 9600 Baud, Keine Parität, 8 Daten-Bits, RTS/CTS und XON/XOFF aus, das High Bit ist zwar angeschaltet, wird aber wegen der Einstellung 8 Daten-Bits nicht beachtet. Das High Bit ist nur bei weniger als 8 Daten-Bits relevant.

## Druckeranpassung

#b123456

| | |
|---|---|
| 1. Typ | 0=Matrix, 1=Typenrad |
| 2. Farbe | 0=S/W, 1=Farbe |
| 3. Punktdichte | 0=1280, 1=960 |
| 4. Qualität | 0=Draft, 1=NLQ |
| 5. Port | 0=Centronics, 1=RS232 |
| 6. Papier | 0=Endlos, 1=Einzelblatt |

Auch der Drucker ist in unserer Desktop-Info ganz normal konfiguriert als Schwarzweiss-Matrixdrucker mit 1280 dpi, Draft-Qualität und Endlospapier. Er ist angeschlossen am Centronics-Port.

## Kontrollfeld

#c777...411H1H2

Die ersten 48 Ziffern stellen die 16 verfügbaren Farben dar. Dabei werden immer drei Ziffern für eine Farbe benutzt.

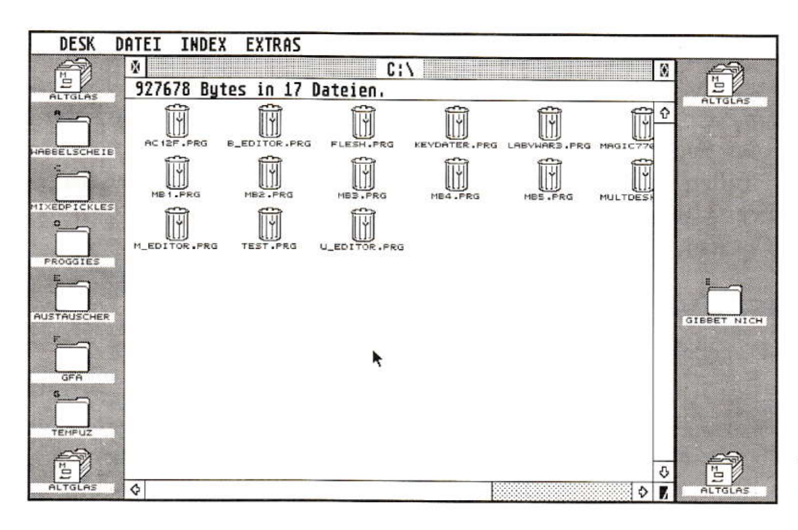
```
#a000000
#b000000
#c00077770007000700552005055522
      20770557075055507706011103
#d
#Z 01 C:\PECSHELL\PECSHELL.PRG@
#E F8 13
#W 00 00 14 01 21 18 0C C:\*.PRG@
#W 00 00 38 01 18 18 00 @
#W 00 00 0E 01 10 18 00 @
#W 00 00 22 01 23 18 00 @
#M 00 05 01 FF G TEMPUZ@ @
#M 00 04 01 FF F GFA@ @
#M 00 03 01 FF E AUSTAUSCHER@ @
#M 00 02 01 FF D PROGGIES@ @
#M 00 01 01 FF C MIXEDPICKLES@ @
#M 00 00 01 FF A WABBELSCHEIB@ @
#M 07 04 01 FF B GIBBET NICH@ @
#T 07 07 00 FF M ALTGLAS@ @
#T 07 00 00 FF M ALTGLAS@ @
#T 01 00 00 FF M ALTGLAS@ @
#T 01 06 00 FF M ALTGLAS@ @
#F FF 04    @ *.*@
#D FF 01    @ *.*@
#G 02 FF    *.APP@ @
#G 02 FF    *.PRG@ @
#P 02 FF    *.TTP@ @
#F 02 04    *.TOS@ @
#F 02 04    *.PEC@ @
#G 03 04    F:\GFA.30\M.
            PRG@ *.GFA@
#G 03 04    G:\TEMPUS\TEMPUS.
            PRG@ *.DUP@
#G 03 04    G:\TEMPUS\TEMPUS.
            PRG@ *.TXT@
#G 03 04    G:\TEMPUS\TEMPUS.
            PRG@ *.INF@
#G 03 04    C:\MULTDESK.PRG@
            *.AC?@
```

*Bild 1: Eine manipulierte DESKTOP.INF-Datei*

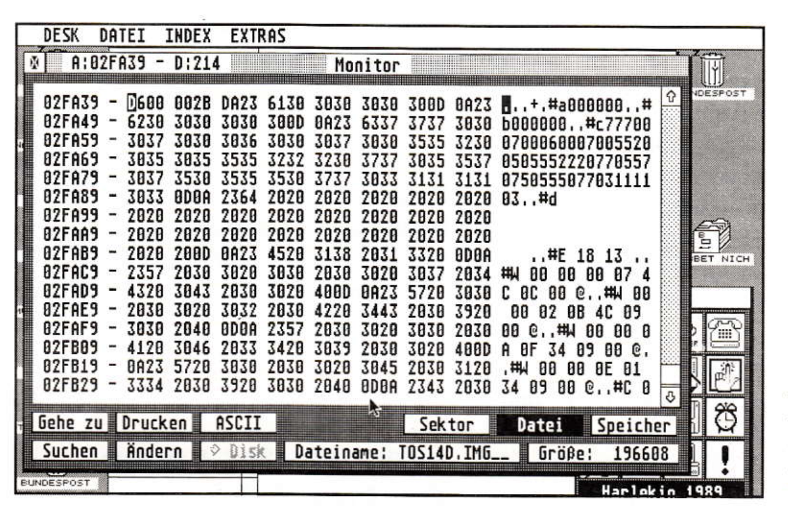
Die Ziffern haben eine Wertigkeit zwischen 0 und 7. Sie stellen die Intensität der jeweiligen Grundfarben Rot, Grün und Blau dar.

| | |
|---|---|
| 49. Maus-Speed | 0 bis 4 |
| 50. Tastenklick | 0=Aus, 1=An |
| 51. Glocke | 0=Aus, 1=An |
| H1. Wiederholzeit | |
| H2. Wiederholrate | |

Unser Desktop-Hintergrund wird invertiert dargestellt, also weiß auf schwarz statt schwarz auf weiß. Die Wiederholrate der Tasten steht auf 3, die Wiederholzeit (also die Zeit, die der ST abwartet, bis er die Taste bei dauerndem Drücken das zweite Mal auf den Bildschirm bringt) beträgt Hex 11, also 17. Die Glocke ist an, aber der Tastaturklick ist ausgeschaltet. Die Maus bewegt sich mit der Geschwindigkeit 6. Eigentlich kann hier nur ein Wert zwischen 0 und 4 stehen, und die 6 wirkt sich auch keinesfalls vorteilhaft auf die Geschwindigkeit der Maus aus, dafür haben wir aber ein individuelles Desktop-Info.




*Das Ergebnis unserer manipulierten DESKTOP.INF-Datei*




*Ändern der Desktop-Info direkt in der Datei TOS.IMG*

## Darstellung

#E H1 H2

H1-Bits

| | |
|---|---|
| 7. Index als | 0=Bild, 1=Text |
| 6/5. Sortiert nach | 00=Name, 01=Datum, 10=Größe, 11=Typ |
| 4. Löschbestätigung | 0=Nein, 1=Ja |
| 3. Kopierbestätigg. | 0=Nein, 1=Ja |

H2
01 = 320*200 oder monochrom
02 = 640*200 oder monochrom
03 = monochrom oder 320*200

Die Dateien werden angezeigt als Text und sortiert nach Typ. Lösch- und Kopierbestätigung sind eingeschaltet, damit nichts unbeabsichtigt verloren geht. Außerdem steht die Auflösung (im zweiten Bit) auf monochrom oder 320*200, ersteres trifft hundertprozentig zu.

## Fenster

#W H1 H2 H3 H4 H5 H6 H7 NAME

H1: horizontaler Schieber
H2: vertikaler Schieber
H3: x-Position
H4: y-Position
H5: Breite
H6: Höhe
H7: vertikales Rollfeld

Ist ein Name vorhanden, wird das entsprechende Fenster geöffnet und das entsprechende Verzeichnis von Diskette bzw. Platte geladen und angezeigt. Interessanterweise kann man sich auch nur alle Programme anzeigen lassen, dazu muß das "*.*" gegen ein "*.PRG" ausgetauscht werden. Genau das ist hier auch geschehen. Horizontaler und vertikaler Schieber stehen auf 0. Unser Fenster beginnt bei der Position 14,1 (ganz oben in der Mitte), ist 21 Einheiten breit und 18 Einheiten hoch. Eigentlich sind nur 17 Einheiten möglich, aber durch diese Manipulationen reichen alle Fenster noch eine Zeile tiefer. Im Fenster werden alle Programme aus dem Wurzelverzeichnis von C:\ angezeigt. Die Position bzw. die Breite und Höhe der Fenster errechnet sich folgendermaßen: Die x-Position/Breite bei H3/H5 wird mit 8 multipliziert, während die y-Position/Breite H4/H6 bei monochrom mit 16 und bei Farbe mit 8 multipliziert wird.

## Diskstations

#M H1 H2 H3 H4 D INCONNAME

H1: x-Position
H2: y-Position
H3: Bild des Icons:
00=Diskstation, 01=Ordner, 02=Müll, 03=Programm, 04=Datei
H4: ?
D: Laufwerksbezeichnung (A-P)

In diesem Desktop sind also 7 Laufwerke mit den Namen *Tempuz* (Textverarbeitung, Laufwerk G), *GFA* (Programmiersprachen, Laufwerk F), *Austauscher* (Zwischenlager für alles mögliche, Laufwerk E), *Proggies* (alle Arten von Programmen, Laufwerk D), *Mixedpickles* (querbeet, Laufwerk C), *Wabbelscheib* (Laufwerk A) und *Gibbet nich* (Laufwerk B, das es wirklich nicht gibt). Sie alle, bis auf Laufwerk B, befinden sich an der x-Position 0 und stehen in Reih' und Glied untereinander.

Die x-Positionen der Icons müssen mit 80, die y-Positionen mit 40 (bei Farbe 20) multipliziert werden, um die wirkliche Bildschirmposition zu bekommen. Zu der y-Position müssen zusätzlich noch 20 Pixel dazugerechnet werden, die die Menüleiste in Anspruch nimmt. Alle Laufwerke haben das Icon eines Ordners (öfter mal was Neues).

## Papierkorb

#T H1 H2 H3 H4 ICONNAME

H1: x-Position
H2: y-Position
H3: Bild des Icons:
00=Diskstation, 01=Ordner, 02=Müll, 03=Programm, 04=Datei
H4: ?

Die Papierkörbe haben hier die Form eines Laufwerks und die Bezeichnungen *Altglas*. Außerdem befindet sich in jeder Bildschirmecke ein Papierkorb, damit dieser von jeder Position aus gut zu erreichen ist (lange Fahrwege mit der Maus kosten Zeit und Nerven).

## GEM-Anwendungen

#G H1 H2 Anwendung Dokument

H1: Bild des Icons:
00=Diskstation, 01=Ordner, 02=Müll,
03=Programm, 04=Datei
H2: ?

Anwendung: Alle .PRG-Dateien sind (normalerweise) als GEM-Anwendungen angemeldet. Hier sind es PRG- und APP-Dateien. Das Dokument ist eine Datei, bei dessen Aufruf automatisch das Programm mit dem Namen, der bei Anwendung steht, vorab geladen wird. Im Desktop kann man das dadurch erreichen, daß man das vorab zu ladende Programm einmal anklickt, im Optionsmenü *Anwendung anmelden* selektiert und jetzt bei Dokument Art den entsprechenden Dateityp einträgt. Auf diese Weise wird zum Beispiel Tempus bei TXT-, DUP- und INF-Dateien nachgeladen, das Programm M bei GFA-BASIC-Programmen und Multidesk bei allen Accessories. Alle GEM-Programme werden als Mülleimer dargestellt.

## TOS-Anwendungen

#F H1 H2 Anwendung Dokument

Wie *#G*, jedoch für TOS-Anwendungen. Alle .TOS-Dateien sind (normalerweise) als TOS-Anwendungen angemeldet. Im Desktop-Info sind auch alle *.PEC-Dateien als solche definiert, das hat allerdings keinen bestimmten Grund, sondern dient nur als Scherz. Alle TOS-Programme werden ebenfalls als Mülleimer dargestellt.

## TTP-Anwendungen

#P H1 H2 Anwendung Dokument

Wie *#F*, nur mit Parameterübergabe. Alle .TTP-Dateien sind (normalerweise) als TTP-Anwendungen angemeldet. Auch diese Dateien werden als Mülleimer dargestellt.

#Z H1 Programmname

H1: 00=TOS-Programme,
01=GEM-Programme

Ab TOS-Version 1.4 existiert auch dieser Parameter. Er gibt ein Programm an, das automatisch nach dem Booten gestartet werden soll - das kann auch ein GEM-Programm sein. In unserem Desktop-Info ist das das Programm *PECSHELL.PRG*, das ein CLI-Programm darstellt. Es wird automatisch nachgeladen und gestartet.

## Brennen des neuen Desktop-Infos

Wenn Sie mit Ihrem neuen Desktop zufrieden sind und nicht immer von einer ganz bestimmten Diskette booten möchten, um die veränderten Einstellungen einzuladen, können Sie natürlich auch alle Änderungen in Ihr TOS im ST einbauen. Dazu benötigen Sie lediglich ein Monitorprogramm, mit dem Sie in einer Datei etwas ändern können (etwa den SED aus Scheibenkleister oder Harlekin), und eine TOS.IMG-Datei auf Diskette oder Festplatte. Die TOS.IMG-Datei erhalten Sie mittels des ROMSAVE-Programms aus der Januar-Ausgabe. Suchen Sie einfach mittels des Monitors nach der Zeichenkombination *#W* oder *#a*, sie steht recht weit am Ende der TOS.IMG-Datei. Hier steht eine ganz normale DESKTOP.INF-Datei, die benutzt wird, wenn auf der Diskette bzw. Platte selbst keine solche vorhanden ist. An dieser Stelle können Sie nun nach der oben beschriebenen Methode die Daten ändern, die Sie für wichtig ansehen. Wenn Sie damit fertig sind, nehmen Sie die TOS.IMG-Datei und brennen sie gemäß der Anleitung in Ausgabe 1/90 und 2/90 in ein EPROM. Wenn Sie das nächste Mal Ihren ST booten, wird er sich mit den von Ihnen geänderten Einstellungen melden, sofern sich keine andere DESKTOP.INF auf Diskette oder Platte befindet. Noch ein Tip am Rande: Sollten Sie nicht genau wissen, was Sie ändern müssen, speichern Sie einfach einmal die Original-DESKTOP.INF-Datei mittels *Arbeit sichern* aus dem Desktop ab und benennen Sie um. Nehmen Sie dann Ihre neuen Einstellungen vor und speichern Ihre neue DESKTOP.INF-Datei wie zuvor ab. Wenn Sie jetzt beide Dateien in einem Editor vergleichen, wissen Sie, was Sie ändern müssen.

Nun kann ich Ihnen eigentlich nur noch viel Spaß mit Ihrem veränderten ST wünschen. Ob rot, ob grün, ob blau: Das Desktop bleibt nicht grau!

*MP*

# Kaum zu glauben ...

## AEGIS-Animator - nun PD

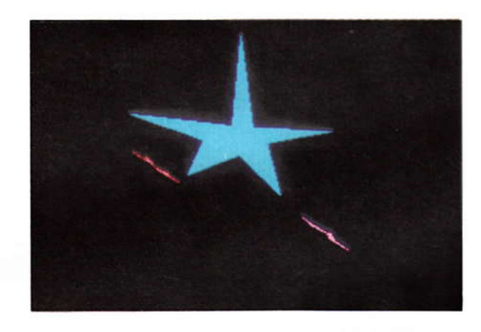
**Wollten Sie schon immer eigene Animationen erstellen? Haben Sie sich mitunter auch geärgert, daß Ihr ATARI ST in bezug auf Grafik dem COMMODORE AMIGA ein wenig unterlegen ist und es für den AMIGA mehr Animations-Software gibt? Seit kurzer Zeit ist der AEGIS-ANIMATOR für den ATARI ST auf PUBLIC DOMAIN-Diskette Nr. 256 zu haben. Weil aus rechtlichen Gründen aber nur das Programm selbst ohne irgendeine Dokumentation und ohne jegliche Demos als PD ausgeliefert werden darf, möchten wir Ihnen mit diesem Artikel eine Anleitung an die Hand geben, eigene Animationen zu erstellen.**

### Hardware-Voraussetzungen

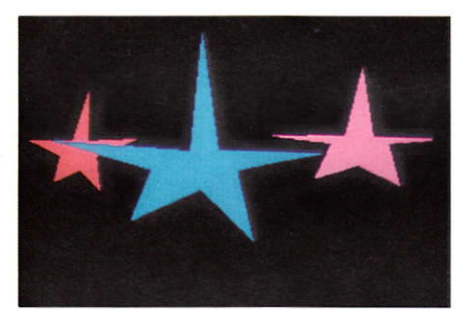
Der AEGIS-Animator läuft auf allen ATARI ST-Computern, die mindestens 512 kByte besitzen. Sollte jedoch nur ein halbes Megabyte in Ihrem Computer eingebaut sein, ist ein TOS im ROM unbedingt nötig, obwohl der Speicher auch dann sehr begrenzt ist. Mit einem Megabyte läßt sich sehr effektiv arbeiten. Außerdem muß ein Farbmonitor vorhanden sein oder ein entsprechender Multi-Sync, da das Programm nur in der niedrigen Bildschirmauflösung (320 x 200 Punkte) läuft. Die Ansteuerung mehrerer Diskettenlaufwerke und einer Festplatte funktioniert problemlos.

### Animationsarten

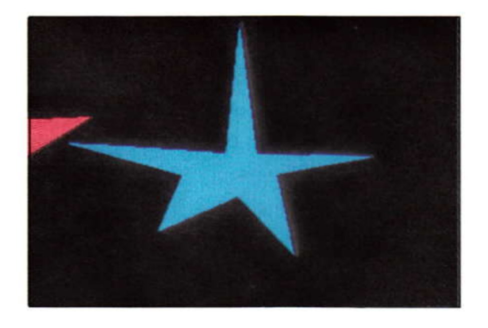
Der AEGIS-Animator stellt dreierlei Arten von Animation zur Verfügung. Dies sind die Colorcylce-, die Celophane- sowie die Metaphoric-Animation. Den ersten Typ kennen Sie bestimmt schon von Malprogrammen wie DEGAS oder NEOCHROM. Bei dem Fachausdruck "Colorcylce" handelt es sich um nichts anderes, als um das "Durchschalten" von einzelnen Farben in einer vorgegebenen Reihenfolge - so wie beim bekannten Bild vom "Wasserfall", bei dem die verschiedenen Blautöne im Kreis durchgewechselt werden, wodurch der Eindruck fließenden Wassers erzeugt wird.











Eine sehr interessante Möglichkeit bietet der Animator durch die "Cel-" oder "Celophane-"Animation. Hierbei wird eine ähnliche Technik verwendet, wie sie bei der Erstellung von Zeichentrickfilmen zur Anwendung kommt: Einzelne Körperteile einer Figur (die sich bewegen oder verändern soll) werden auf durchsichtige Folie gemalt. Von Bild zu Bild werden diese Körperteile geringfügig bewegt oder durch ähnliche ersetzt.

Ein Beispiel zur Verdeutlichung: Micky Maus möchte sich als Hampelmann sportlich betätigen. Hierzu erstellt der Zeichner aus Folie einzelne Körperteile wie Rumpf, Kopf, Unter- und Oberarme sowie Unter- und Oberschenkel. Eventuell kommen weitere Kopfteile hinzu, bei denen die Figur lächelt oder etwa mit den Augen blinzelt. Danach legt der Zeichner zunächst Rumpf und Körperteile in Grundstellung zusammen und filmt ein Bild ab. Als nächstes bewegt er die Körperteile ein wenig und belichtet ein weiteres Bild. So "animiert" er die Maus dazu, eine Hampelmannbewegung auszuführen und wechselt zwischendrin zum Beispiel den Kopf für ein paar Bilder durch einen zweiten aus, der mit einem Auge blinzelt usw., bis aus den einzelnen Bildern schließlich eine ablauffähige Sequenz entstanden ist. Läßt man dann die Bilder mit einer Geschwindigkeit von mindestens 24 Bildern pro Sekunde ablaufen, erscheint unserem Auge die Animation als fließende Bewegung. Bekanntere Filme, wie etwa "Asterix - Operation Hinkelstein", entstanden nach genau diesem Prinzip. Das Wort "Celophane" bezieht sich auf die "Folien"technik, und nicht anders arbeitet diese Art der Animation. Nur handelt es sich in Ihrem Computer nicht um einzelne Folienteile, sondern um Pixelgrafiken, die Sie zusammensetzen und bewegen oder durch andere auswechseln.

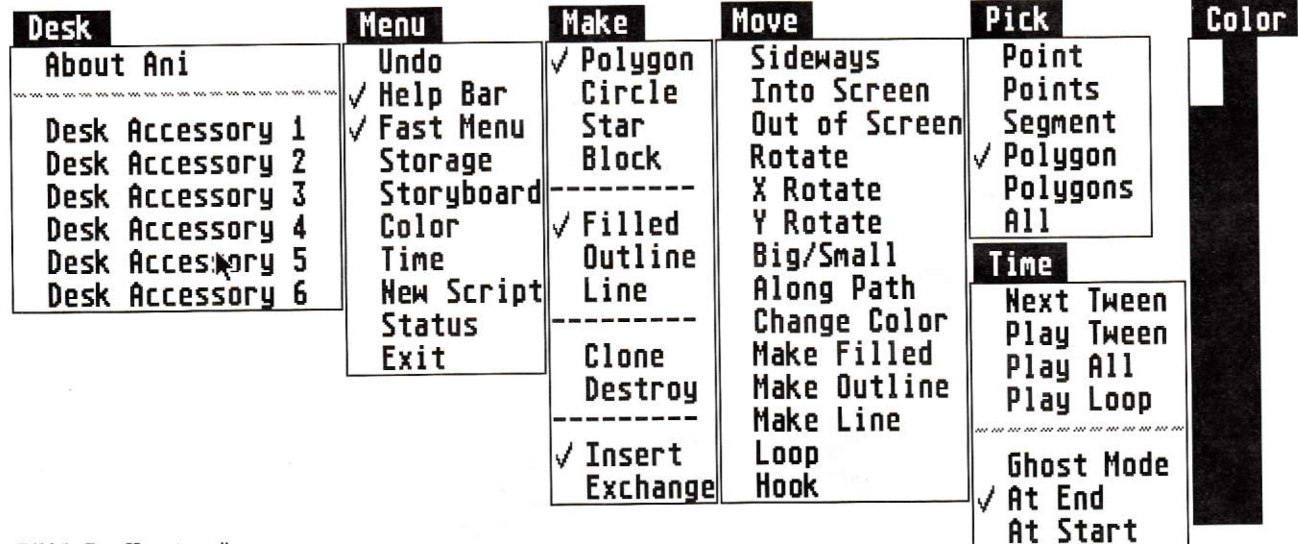



*Bild 1: Das Hauptmenü*

Die dritte Animationsart nennt sich "METAPHORIC"-Animation. Das Animationsprogramm verwaltet einfachere geometrische Vektorfiguren, die sich um die drei Koordinatenachsen drehen lassen. Bewegen in x-, y- und z-Richtung ist ebenso selbstverständlich, wie Verkleinern, Vergrößern und Verändern der geometrischen Form. Der Animator bietet fünf Grundfiguren an. Dies sind Polygon, Kreis, Stern (mit fünf Ecken), Viereck und Linie. Durch zahlreiche Zusatzfunktionen lassen sich diese (bis auf das Viereck) aber in beliebige Objekte verwandeln oder verzerren. Außerdem haben Sie die Wahl zwischen gefüllten oder leeren Objekten.

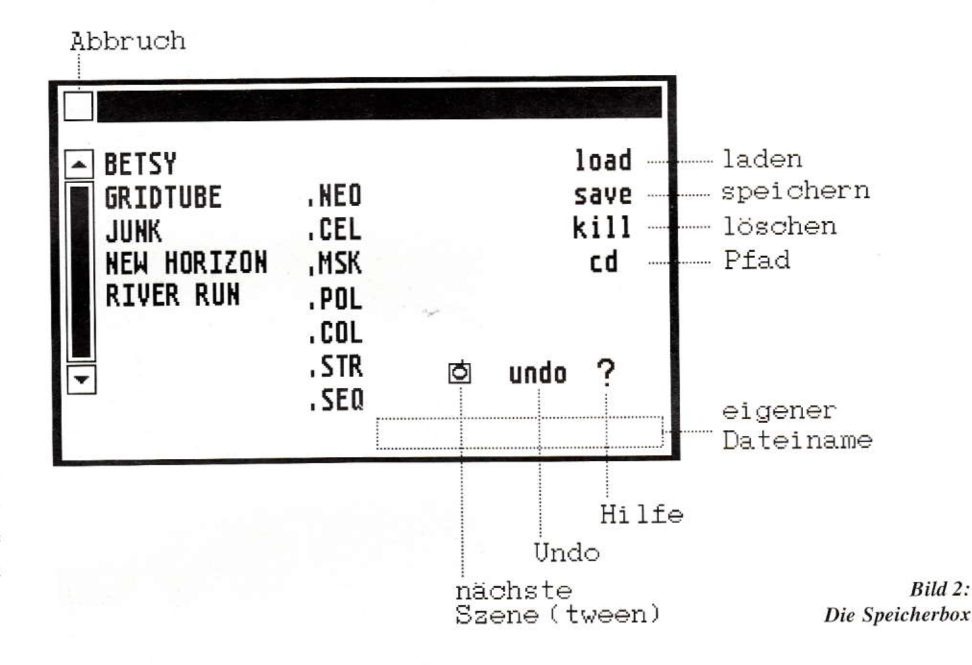



***Bild 2: Die Speicherbox***

## *"Menu"*

Ein Programm auf der Diskette ist der sogenannte "PLAYER", der nur dem Abspielen von fertigen Filmen dient. Er durfte schon früher beliebig kopiert und weitergegeben werden. Seit das Programm selbst allerdings auch PD ist und vervielfältigt werden darf, wird der Player nicht mehr benötigt. Deshalb beschränken wir uns in diesem Artikel auf den Animator selbst.

Starten Sie das Animationsprogramm durch Doppelklick in niedrigster Auflösung, und nach kurzer Zeit müßte das Hauptmenü erscheinen (siehe Bild 1). Zunächst wollen wir Ihnen die einzelnen Menüpunkte erläutern. Demos zur Anregung finden Sie auf der PD-Diskette Nr. 256. Der Eintrag "Desk" erklärt sich von selbst und muß wohl nicht näher erläutert werden.

Interessant wird es für Sie unter "Menu". Hier finden Sie den Eintrag "Undo", der zum Rückgängigmachen der letzten Aktion dient. Haben Sie also einmal aus Versehen ein Objekt bewegt und wollen diese Bewegung ungeschehen machen, benutzen Sie dazu den Menüpunkt "Undo". Der nächste Eintrag lautet "Help Bar". Hiermit können Sie die Hilfszeile - die am unteren Bildschirmrand eingeblendet wird - ein- oder ausschalten. Ebenso verhält es sich mit dem Menüeintrag "Fast Menu". Damit können Sie ein kleines grafisches Menü, das als Alternative zum PULL-DOWN-Menü angeboten wird, erscheinen oder verschwinden lassen.

Der Punkt "Storage" ist das, was in deutschen Programmen meist unter "Datei" zu finden ist. Hiermit werden sämtliche Diskettenoperationen durchgeführt, Sie können also Laden, Speichern, Löschen usw. Wählen Sie "Storage", erscheint eine Auswahlbox (Bild 2), die rechts oben die Punkte "load", "save", "kill" und "cd" anbietet. Auf der linken Seite wählen Sie den Dateinamen aus, und in der Mitte dieser Box haben Sie die verschiedenen Extensions, die Ihnen das Programm anbietet. Dabei handelt es sich bei ".NEO" um Neochrome-Bilder, bei ".CEL" und ".MSK" um einzelne Folienteile, ".POL" sind Polygone und ".COL" Farben. Die Extensions: ".STR" steht für Strip beziehungsweise für Filme, ".SCR" für Script, also sozusagen für das Drehbuch. Wenn Sie einen 'Strip' mit der Endung ".STR" laden, werden automatisch alle für diesen Film benötigten Dateien hinzugeladen. Mittels "load" können Sie Dateien einladen, mit "save" speichern und durch "kill" von der Diskette löschen. Die Funktion "cd" dient dazu, einen neuen Pfad per Tastatur einzugeben und neu festzulegen.

Die Unteraktion "UNDO" wirkt als Abbruchkommando, das Fragezeichen "?" gibt einen Hilftext aus, nachdem ein

weiteres Menü angewählt wurde. Mittels des Kästchens mit dem darin enthaltenen Kreis können Sie auf den nächsten "Tween" umschalten, also auf die nächste Filmszene vorschalten (Funktion siehe unten).

Durch die Funktion "Storyboard" läßt sich auf Menü zwei umschalten, das benötigt wird, wenn Sie mehrere Filme aneinanderhängen möchten (Bild 5, Erklärung siehe unten).

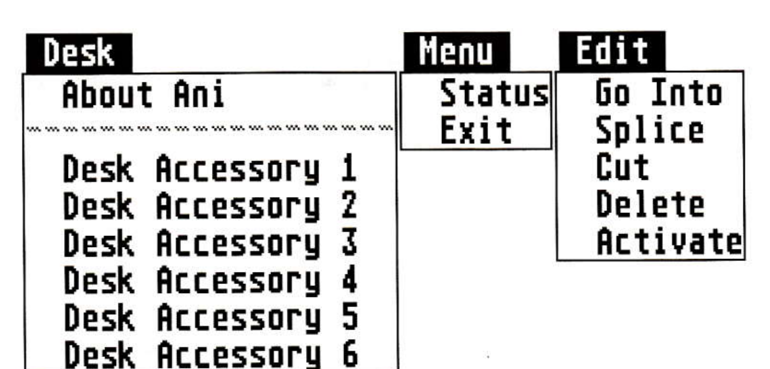



*Bild 3: Es läßt sich auf eine zweite Menüleiste umschalten.*

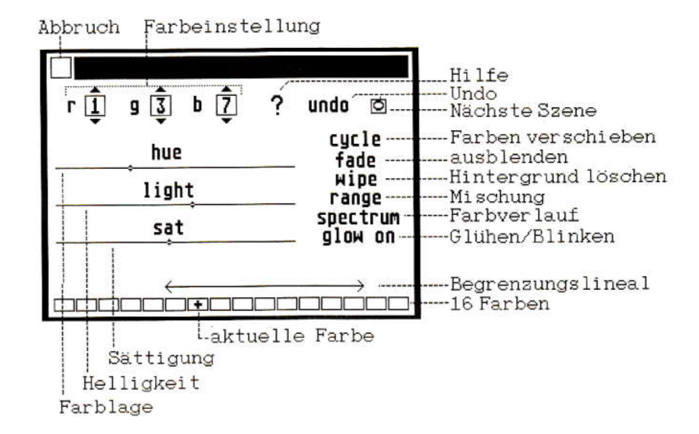



*Bild 4: Die Farbauswahlbox*

## *Farbmanipulation*

Durch den Menüeintrag "Color" können Sie bestimmen, wie die Farbpalette aussehen soll. Hierzu suchen Sie aus Bild vier am unteren Rand eine Farbe aus, die Sie verändern möchten. Mit den R/G/B-Reglern können Sie Ihre Farbe individuell manipulieren. Wissen Sie die Zahlenkombination der Farbe nicht, gibt es auch eine zweite Möglichkeit: Durch die drei Schieberegler "hue", "light" und "sat" bestimmen Sie die Farblage (hue) in der Palette aus 512 möglichen Farben, die Helligkeit beziehungsweise den Schwarzweißanteil (light) und die Farbsättigung/-intensität (saturation). Undo und das Viereck mit enthaltenem Kreis wirken genauso wie bei "Storage".

Eine Vielzahl von Farbmanipulationen bieten die Zusatzkommandos in Bild vier: Mit Hilfe des "Begrenzungslineals" können Sie einstellen, auf welchen Farbenbereich sich Ihre Veränderungen auswirken sollen. "Cycle" verschiebt die Farben im eingestellten Bereich um eine Position nach rechts pro Klick auf Cycle. Haben Sie also das Lineal auf drei Farben eingestellt, wird Farbe 1 zu 3, Farbe 2 zu 1 und Farbe 3 zu 2. "Fade" blendet zwischen zwei Farben um. Hierzu stellen Sie wiederum Ihr Lineal auf die Farben, die verändert werden sollen, und setzen das Kreuz in die Farbe, auf die Sie überblenden möchten. Danach klicken Sie auf Fade, und Ihre ausgewählten Farben werden in dieser Szene "umgeblendet". Durch "Wipe" können Sie einen Hintergrund löschen, den Sie als NEOCHROME-Bild geladen haben.

Mit "Range" können Sie eine Farbmischung aus zwei Farben erzeugen. Schieben Sie dazu Ihr Lineal auf den Farbbereich, den Sie verändern möchten. Am linken und rechten Rand stellen Sie die beiden "Randfarben" ein, die gemischt werden sollen. Durch Anwahl von Range erzeugt Ihnen der Animator einen feinen Übergang - je größer der Bereich, desto feiner die Mischfarben. Wenn Sie also als Randfarben Rot und Grün einstellen, erhalten Sie in der Mitte als Mischfarbe Braun.

"Spectrum" erzeugt einen Farbverlauf zwischen zwei Randfarben. Die Handhabung ist die gleiche wie bei Range, nur entsteht keine Farbmischung, sondern ein Farbverlauf. Zur Verdeutlichung hilft es am besten, auszuprobieren.

## *Szenen / "Tweens"*

In der Filmbranche wird Bild für Bild mit einer sogenannten Einzelbildkamera gefilmt. Von Bild zu Bild treten jeweils kleinere oder größere Veränderungen auf. Sollen einzelne Szenen länger dauern, müssen die Bewegungen von Bild zu Bild feiner sein, was dazu führt, daß ganz genau geplant werden muß, wie lange eine Szene sein soll. Hier macht es uns der Computer wesentlich einfacher. Wir brauchen nur unsere Veränderung zu erzeugen und stellen danach die Zeit ein, die benötigt werden darf.

Somit ist es uns möglich, einzelne Szenen zu erzeugen, in denen mit unseren Objekten etwas Bestimmtes geschieht. Der Animator hält sich übrigens sehr genau an die Zeiten, die ihm für eine Szene vorgegeben werden. Wird ihm die Darstellung zu schnell, wird die Animation nicht etwa langsamer (was bedeuten würde, daß der Zeitplan nicht mehr stimmt), sondern die Darstellung erfolgt etwas ruckhafter.

Ein Beispiel für die Aufteilung in einzelne Szenen: Wir wollen einen Stern von links nach rechts bewegen. Dort soll er sich um 180 Grad drehen und dann an die Ausgangsposition zurückkehren. Hierzu benötigen wir drei Szenen: 1. Bewege von links nach rechts. 2. Drehe um 180° 3. Bewege von rechts nach links.

Jede dieser Einzelszenen heißt für unseren AEGIS-Animator "Tween". Die Parameter eines solchen Tweens lassen sich durch den Menüpunkt "Time" einstellen. Nach Aufruf von "Time" erscheint Bild 5. Jede unserer Szenen erhält eine Nummer, die Numerierung beginnt übrigens bei 0001. Durch die Pfeile bei "which tween" suchen Sie die Szene aus, deren Parameter Sie verändern möchten. Mit "tween time" können Sie die Zeit einstellen, die der Computer für die Szene brauchen darf. Die Zeit ist in Abschnitte von 1/60 Sekunde eingeteilt. Stellen Sie die "tween time" auf 0060, dauert Ihr Tween genau eine Sekunde. Eine Szene kann maximal fünf Sekunden dauern (0300). Wenn Sie im Menü "TIME - Next Tween" auf eine neue Szene umschalten, erhält diese die gleiche "tween time"-Einstellung wie die vorhergehende Szene. Dies sollte berücksichtigt werden, um unnötige Rückstelarbeit zu vermeiden. Dies gilt dann, wenn Sie mehrere Tweens hintereinander erstellen, ohne TIME zu verändern. Durch

den Menüpunkt "global speed" läßt sich die Geschwindigkeit einstellen, mit der der komplette Film ablaufen soll. Der Standardwert hierfür liegt bei 32. Verändern Sie die "global speed", so zeigt der schwarze Balken die neu berechnete Zeit der Szenen bis zu diesem Zeitpunkt an. Die Gesamtdauer des Films läßt sich einfach feststellen, indem Sie "which tween" auf die letzte Szene stellen.

Ihre Arbeit können Sie auch löschen, etwa, wenn Ihnen der eigene Film plötzlich doch nicht gefällt, und Sie eine ganz neue Animation erstellen möchten. Klikken Sie hierzu auf den Menüeintrag "New Script", und die bisher erstellten Szenen werden gelöscht. Durch Anwahl von "Status" erhalten Sie die neuesten Informationen über Ihre Speicherplatzverhältnisse. "Exit" beendet das Programm. Kontrollieren Sie deshalb vorher, ob Sie durch "Storage" Ihre Daten gesichert haben.

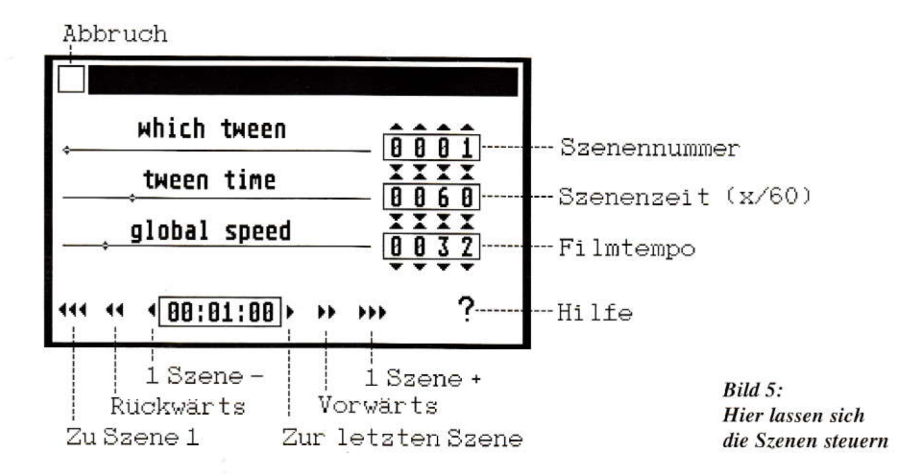



*Bild 5: Hier lassen sich die Szenen steuern*

## "Make"

Im Pulldown-Menü "MAKE" stellen Sie die Art des Objektes ein, das Sie erstellen möchten. Sie haben die Auswahl zwischen "Polygon", einem Gebilde aus (fast) beliebig vielen Linien, "Circle", einem Polygon, dessen Punkte in Kreisform angeordnet sind, "Star", einem Polygon in Sternform mit fünf Ecken, und "Block". Einzige Ausnahme ist dieser "Block", denn er ist kein Polygon aus Linien, sondern ein festes Viereck. Versuchen Sie also etwa, ein beliebiges Polygon und einen Block um die z-Achse zu drehen, erhält das Polygon schiefe Linien und dreht sich. Der Block jedoch bleibt in seiner Form erhalten und bewegt sich nur kreisförmig um den Drehpunkt. Zum besseren Verständnis hilft es, "Rotate" mit einem Polygonrechteck aus vier Linien und einem Block auszuprobieren.

Als Parameter haben Sie die Wahl zwischen "Filled" (gefüllt), "Outline" (nur die Linien, Endpunkte verbunden) und "Line" (nur die Linien, Endpunkte nicht verbunden).

Der Menüpunkt "Clone" dient dazu, einzelne oder mehrere Objekte zu kopieren. Wenn Sie zum Beispiel einen Stern haben und hieraus drei Stück entstehen lassen möchten, schalten Sie auf "Clone" und klicken zweimal auf den Stern. Daß Sie zwei neue Sterne erzeugt haben, sehen Sie, wenn Sie den Stern bewegen. Zwar erscheint in der Hilfszeile "Release Clone At Place Of Birth", dieser Programmpunkt funktioniert jedoch nicht immer einwandfrei. Meist erscheint die Kopie an der gleichen Stelle wie das Original.

Falls Sie mehrere Objekte erzeugt haben, möchten Sie bestimmt auch mal eines oder mehrere Ihrer Gebilde vom Bildschirm löschen. Hierzu dient der Menüpunkt "Destroy". Vorsicht! Wenn Sie im Menü "Make" auf "Destroy" schalten, wird das Polygon, das Sie anwählen, gelöscht. Das Löschen tritt bereits am Anfang einer Szene in Kraft und nicht erst am Ende. Wollen Sie also ein Element erst drehen und dann löschen, müssen Sie in der aktuellen Szene drehen und dürfen erst im nächsten Tween mittels "Destroy" löschen!

"Insert" und "Exchange" beziehen sich auf das Laden von Objekten. Stellen Sie auf "Insert", und laden Sie ein Objekt durch "Storage". Positionieren Sie das Bild, und lassen Sie den Mausknopf los. Das geladene Objekt wird nun in die Animation eingefügt. Etwas anders funktioniert "EXCHANGE". Laden Sie wiederum ein Bild von Diskette oder Festplatte. Positionieren Sie das Objekt auf einem anderen bereits vorhandenen Objekt, und lassen Sie den Mausknopf los. Das alte Element wird nun gelöscht und durch das neu geladene ersetzt.

## "Move"

Zweifelsohne ist das interessanteste Menü des Animators der Menüpunkt "MOVE", durch den Sie sämtliche Bewegungen und Aktionen durchführen können. Durch "Sideways" bestimmen Sie, daß es sich um eine seitliche Bewegung handelt. Nehmen Sie durch Halten der Maustaste ein Objekt auf, und bewegen Sie es. Wenn Sie die Taste loslassen, wird das Objekt an die neue Position gesetzt. Wenn Sie durch "TIME - next tween" auf eine neue Szene umschalten, bewegt sich das Objekt in dieser Szene vom Anfangspunkt zu dem Punkt, zu dem Sie es "geschoben" haben.

Mit den beiden Menüpunkten "Into Screen" und "Out of Screen" können Sie Objekte entlang der z-Achse in den Bildschirm hinein oder heraus bewegen. Diese Bewegung sieht im ersten Moment wie eine Verkleinerung aus, macht sich aber erst bei einer Drehung richtig bemerkbar. Merken Sie sich nach Möglichkeit, wie weit Sie ein Objekt hinein- oder herausbewegt haben, oder notieren Sie dies kurz. Möchten Sie die räumlich verschobene Grafik in einer späteren Szene nämlich mit anderen Polygonen auf einer Ebene rotieren, müssen Sie diese ja in gleicher z-Koordinate positionieren.

Drehen von Objekten ist problemlos möglich. Durch die drei angebotenen Funktionen "Rotate", "X Rotate" und "Y Rotate" kann zwischen den Koordinatenachsen gewählt werden. Hierzu erscheint zunächst ein neuer Maus-Cursor, der die Drehachse anzeigt. Der Animator fordert Sie auf, den Drehpunkt festzulegen, um danach mittels Mausbewegung (um den eingegebenen Punkt in etwas Abstand kreisen!) die Drehweite zu bestimmen.

Durch "Big/Small" können Sie Ihre grafischen Objekte vergrößern oder verkleinern. Setzen Sie dazu den Bezugspunkt nicht direkt auf Ihr Objekt, sondern ein wenig entfernt davon. Wenn Sie sich nun vom Objekt auf den Punkt zu bewegen, wird die Grafik verkleinert. Bewegen Sie sich vom Punkt weg zum Objekt hin (bzw. darüber hinaus), findet eine Vergrößerung statt.

Die "Move-Sideways"-Funktion merkt sich den Anfangspunkt und den Endpunkt und führt dann eine direkte Bewegung

durch. Ob Sie Ihr Objekt "auf Umwegen" oder direkt hierhin bewegen, spielt dabei keine Rolle. Wollen Sie aber eine Bewegung im Dreieck durchführen, müssen Sie drei Szenen erstellen, bei denen jeweils in eine Ecke bewegt wird. Um dies zu vereinfachen, existiert "Along Path". Hierbei handelt es sich um eine relative Bewegung zum Anfangspunkt. Der Computer gibt Ihnen eine Linie (einen "Pfad") vor, auf der Sie ein Polygon bewegen können. Der Anfangspunkt der Linie muß aber nicht mit dem Anfangspunkt des Polygons übereinstimmen. Damit lassen sich auch Bewegungen aus dem Bildschirm heraus und wieder hinein problemlos realisieren. Sitzt Ihr Objekt etwa am linken Bildrand, und Sie beginnen Ihren Linienpfad von rechts nach links und zurück, bewegt sich das Polygon ebenso von rechts nach links, verläßt (entlang des Pfades) den sichtbaren Bereich und kehrt danach auf den Schirm zurück. Sie sollten jedoch beachten, daß diese Bewegung entlang eines Pfades im Gegensatz zu "MOVE - Sideways" zeitlich schneller erfolgt.

Durch den Menüeintrag "Change Color" können Sie einem Objekt eine neue Farbe geben. Hierzu wählen Sie erst eine neue Farbe aus dem COLOR-Menü (ganz rechts) und klicken dann auf "CHANGE COLOR". Danach werden Sie aufgefordert, die Grafik anzuwählen, die die neue Farbe erhalten soll.

Durch die drei Menüpunkte "Make Filled", "Make Outline" und "Make Line" können die Parameter eines Objektes verändert werden (siehe auch unter "Make").

Durch "Loop" können einzelne oder mehrere Punkte eines Polygons verzogen und so das Objekt verzerrt werden. Damit lassen sich vor allem kleine Punktfehler nach einer Drehung ausgleichen. Wenn Sie zum Beispiel ein Viereck um mehrere Achsen gleichzeitig drehen, kommt es schon einmal vor, daß eine Ecke danach um einen oder zwei Punkte verschoben ist. Mittels "LOOP" können Sie hier eingreifen und nach der Drehung die Ecke wieder "hinbiegen". Dies fällt im fertigen Film gar nicht auf, da "LOOP" ja innerhalb der selben Szene gleichzeitig mit der Drehung ausgeführt wird.

Die mit Abstand interessanteste Funktion ist "Hook". Hiermit lassen sich nämlich Punkte eines Polygons löschen oder neue

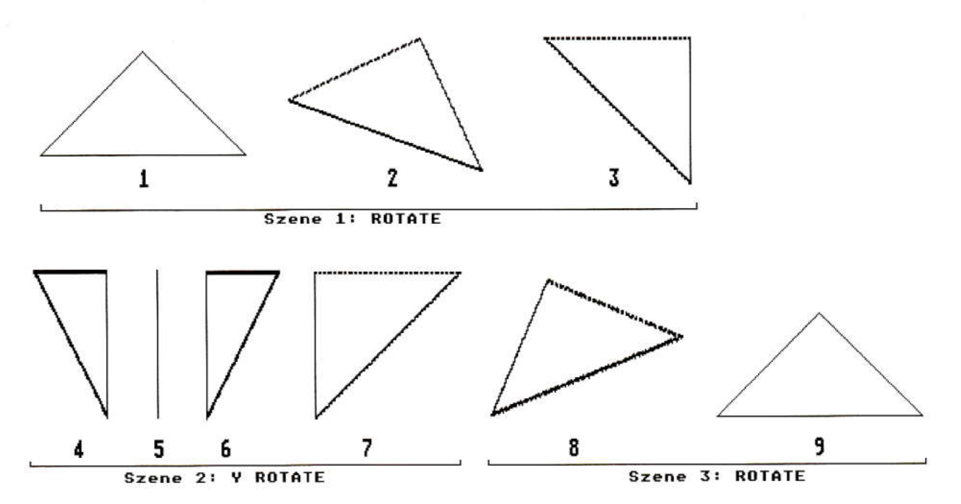



*Bild 6: Das Arbeiten mit "Rotate"*

hinzufügen. Durch geschickte Variation der Polygonform wird aus einem Achteck plötzlich ein Batman usw. Eine ähnliche Technik wird mitunter im Fernsehen angewendet, wenn zwischen zwei Bildern umgeblendet wird: da wird aus einem Rechteck (aktuelles Bild) plötzlich ein Auto, das davonfährt und im Hintergrund das neue Bild zurückläßt usw.

Hierzu wählen Sie "Hook" und suchen anschließend das gewünschte Polygon aus. Indem Sie auf eine Linie zwischen zwei Punkten klicken, erhalten Sie an dieser Stelle einen zusätzlichen Punkt. "Nehmen" Sie einen vorhandenen Punkt mittels Maustaste, so können Sie diesen genauso wie bei "LOOP" bewegen und Ihr Objekt so verzerren. Wenn Sie ihn auf einen anderen vorhandenen Punkt ziehen, schnappt dieser irgendwann auf den festen Punkt ein, und Sie haben den beweglichen Punkt gelöscht. Mitunter fällt dieser gelöschte Punkt nicht ganz exakt mit dem zweiten Punkt zusammen, was sich aber nicht ändern läßt. Außerdem ist die Differenz meist sehr gering.

## "Pick"

Im fünften Menü namens "Pick" können Sie einstellen, auf welchen Bereich sich Ihre Funktionen und Veränderungen beziehen sollen. Möchten Sie nämlich beispielsweise nicht nur EIN Objekt in x-Richtung bewegen, sondern auch ein zweites parallel dazu, können Sie dies unter "PICK" einstellen.

Hier suchen Sie nämlich aus, was Sie verändern möchten: mit "Point" bewegen Sie nur einen einzelnen Punkt, mit "Points" können Sie mehrere Punkte anwählen und durch "Segment" einen Teilabschnitt eines Polygons. Außerdem haben Sie die Wahl, ob sich Ihre Aktionen auf ein einzelnes Polygon ("Polygon"), auf mehrere Objekte ("Polygons") oder auf alle Grafiken beziehen sollen ("All").

## "Time"

Was in der Zeichentricktechnik durch Einzelbildschaltung verwirklicht wird, realisieren wir über die beschriebenen "Tweens" (engl.: Szenen). Durch "Next Tween" schalten Sie auf eine neue Szene um. Mit "Play Tween" können Sie sich die Szene ansehen, die Sie gerade bearbeiten. "Play All" spielt alle bisher erstellten Szenen von Anfang an ab. Möchten Sie ab einer bestimmten Szene die folgenden betrachten, so stellen Sie im Menü "Menu-Time" die gewünschte Szene (=which tween) ein und klicken danach auf "Play Loop". Findet der Animator das Ende des Drehbuches, startet er jedesmal wieder von vorne (Tween 0001), solange, bis Sie die rechte Maustaste drücken.

Im Normalfall werden Sie hauptsächlich gefüllte Objekte benutzen. Falls sich nun zwei Objekte überlagern (zum Beispiel ein Pacman-Polygon mit einem Kreis als Auge), wird etwa bei der y-Drehung irgendwann ein Objekt vom anderen überdeckt. Damit Sie auch die verdeckte Grafik kontrollieren können, gibt es den "Ghost Mode". Hier werden alle Grafiken als Outline dargestellt, womit Sie auch sich überdeckende Polygone sehen können.

Durch die Menüpunkte "At Start" und "At End" können Sie einstellen, ob Sie die Ausgangssituation eines Tweens oder das Ende der Szene sehen möchten.

## Das "Storyboard"

Im sogenannten Storyboard (Bild 5) können Sie verschiedene Filme miteinander verknüpfen oder neue hinzuladen. Der AEGIS-Animator erlaubt, bis zu sechs Filme gleichzeitig im Speicher zu halten (je nach Größe des RAMs) und diese auch aneinanderzuhängen. Das Menü "Desk" erklärt sich wiederum von selbst. Im Pulldown-Menü "Menu" haben Sie ebenfalls die Möglichkeit, sich durch "Status" einen Überblick über den verbleibenden Speicherplatz zu verschaffen.

ACHTUNG - durch Anwahl von "Exit" verlassen Sie nicht etwa das Untermenü Storyboard, sondern Sie beenden den kompletten AEGIS-Animator.

Unterhalb des Pulldown-Menüs sehen Sie sechs Kästchen. Wenn Sie bereits einen Film erstellt haben, wird die erste Szene des Films im linken oberen Kästchen zu sehen sein. Im "Edit"-Menü haben Sie fünf Funktionen zur Verfügung, die Sie auf die maximal sechs Filme im Speicher anwenden können. Mit "Go Into" schalten Sie auf den Film um, auf den Sie mit der Maus klicken, und kehren ins Hauptmenü zurück. Durch den Menüeintrag "Splice" können Sie einen Film an einen anderen anhängen oder kopieren. Klicken Sie zunächst auf "Splice", wählen Sie sodann den Source-Film (=das Fenster) aus, den Sie anhängen beziehungsweise kopieren möchten. Klicken Sie danach auf den Destination-Film (das Fenster), an den Sie anhängen möchten. Ist in diesem Fenster noch kein Film enthalten, wird der ausgesuchte Source-Film hineinkopiert.

Durch die Funktion "Cut" können Sie einzelne Szenen eines Films "herausschneiden" und übertragen oder anhängen. Suchen Sie hierzu mittels der "MENU - Time / which tween"-Funktion im Hauptmenü die entsprechende Szene aus. Wenn Sie bei einem Film aus zehn Szenen beispielsweise auf die fünfte Szene schalten und mit "Cut" arbeiten, so schneiden Sie die 5. bis 10. Szene aus und können diese in ein anderes Fenster übertragen. Im ersten Fenster bleiben dann nur die Szenen eins bis vier zurück. Somit können Sie einzelne Szenen eines Filmes löschen oder in einen anderen an beliebiger Stelle einfügen.

"Delete" funktioniert genauso wie "Menu-New Script". Hiermit können Sie den Film in einem Fenster löschen.

Schließlich bleibt als letzte Funktion "Activate" übrig, womit Sie einzelne Filme im Fenster ablaufen lassen können. Beim ersten Klick starten Sie den Film in einem Fenster, beim zweiten Mausklick halten Sie ihn wieder an. Dadurch können Sie schneller eine bestimmte Szene finden oder nur einfach das Ergebnis eines Zusammenschnittes betrachten.

## Fazit

Es ist kaum zu glauben, daß ein leistungsfähiges Programm wie der AEGIS-Animator nun auf PUBLIC DOMAIN-Basis zu haben ist. Andererseits ist es verständlich, daß das Programm aus rechtlichen Gründen nur ohne Demos und Anleitung als Text-File freigegeben werden durfte. Dies stellt jedoch weniger ein Problem dar, denn die Software läßt sich leicht bedienen, wenn man einmal hinter die Technik der Einzelszenenschaltung mittels mehrerer "TWEENS" gekommen ist. Bereits nach zwei bis drei Stunden des Probierens dürfte es kein Problem mehr sein, eigene Animationen zu erstellen. Die interessanteste Variante ist sicherlich die "Metaphoric-Animation", die die Erstellung von zweidimensionalen Polygonen und eine dreidimensionale Bewegung, Drehung und Manipulation erlaubt. Auf der PUBLIC DOMAIN-Diskette 256 unserer Sammlung liefern wir demnächst auch selbsterstellte Demos mit, und mit Hilfe dieses Artikels sollten Sie mühelos eigene Animationen erstellen können.

*RP*




# Der DFÜ-Dienst der Bundespost

# Bildschirmtext




**Sicher sind Sie auch schon mal über das Wort "Bildschirmtext" gestolpert und wußten am Anfang nichts rechtes damit anzufangen. Text sieht man schließlich jeden Tag auf dem Bildschirm. Das ist aber ganz weit daneben, denn bei Bildschirmtext handelt es sich um ein Dienstleistungsangebot der Deutschen Bundespost. Welche Möglichkeiten Sie durch das System haben (und nicht haben), soll Ihnen dieser Artikel zeigen.**

Da die Deutsche Bundespost momentan mit einem Angebot lockt, das man wirklich überdenken sollte, möchte ich Ihnen in diesem Monat BTX etwas näher bringen. Das Angebot lautet, drei Monate BTX lang testen zu können. Wenn man danach nicht überzeugt ist, kann man seinen BTX-Anschluß zurückgeben und erhält die 65,- DM Anschlußgebühr zurück. Nähere Informationen erhalten Sie an jedem Postamt. Ansonsten kostet Sie BTX lediglich DM 8,- im Monat (plus Telefongebühren und kostenpflichtige Abrufseiten), das sollte auch für einen Schüler leicht aufzutreiben sein.

Was ist überhaupt BTX? Das Bildschirmtext-System der Post ist aufgeteilt in sogenannte Seiten. Auf jeder Seite können sich Informationen, Angebote, Hilfen, Neuigkeiten und andere Dinge verbergen, eben all das, was man auch in einer Mailbox machen kann. In Bildschirmtext existieren zusätzlich zu "normalen" Mailboxen aber auch die Möglichkeit zur Bestellung von Waren, es gibt hübsche Grafiken oder man kann einem Anbieter eine Nachricht zu hinterlassen. Zur weiteren Erklärung des Systems wählen wir BTX einfach mal an.

## Anwahl

Um mit BTX kommunizieren zu können, benötigen Sie zuerst ein Programm, das für diesen Zweck ausgelegt ist, etwa den Btx/Vtx-Manager von Drews oder einen anderen sogenannten "BTX-Decoder". Weiterhin brauchen Sie einen Akustikkoppler, ein Modem oder eine BTX-Anschlußdose der Deutschen Bundespost. Sie können BTX mit 1200/75 Baud in jeder Stadt unter der Nummer (0)190 erreichen, in Großstädten können Sie unter den Nummer 19300 und 19304 auch mit 1200/1200 bzw. 2400/2400 Baud anrufen. Wurden bisher alle Eingaben mit <Return> abgeschlossen, müssen Sie sich beim Bildschirmtextsystem umgewöhnen, denn hier wird jede Eingabe durch ein Doppelkreuz bestätigt.

Nach der Anwahl meldet sich BTX mit einer Blockgrafik. Hier müssen Sie nun Ihre Teilnehmerkennung und das Paßwort eingeben. Wollen Sie das System nur als Gast betrachten, drücken Sie einfach auf Return (bzw. auf das Doppelkreuz, je nach Decoder). Kurze Zeit später wird Ihnen die Hauptauswahl auf dem Bildschirm präsentiert. Angenommen, Sie suchen die aktuellen Musik-Charts, dann ist alles, was Sie einzugeben haben "*30003#" - Diese Seitennummer ist Formel 1 vom Bayerischen Rundfunk (Bild 1).

## Suchfunktionen

Nun wäre es unsinnig, sich so viele Zahlenkombinationen im Kopf behalten zu müssen, weshalb im Bildschirmtextsystem eine Suchfunktion existiert. Vielleicht möchten Sie ja eine Fahrplanauskunft der Deutschen Bundesbahn erhalten. In diesem Fall geben Sie einfach "*BAHN#" ein, um zur entsprechenden Seite zu gelangen (Bild 2). So einfach ist es, eine Zugverbindung von Frankfurt nach Hagen zu erhalten! Anhand der Beispiele haben Sie bestimmt auch schon gesehen, wie das Prinzip der Seitenanwahl funktioniert: Jede Seitenanwahl wird eingeleitet durch einen Stern ("*"), dann folgt die Nummer oder der Name der Seite, zum Schluß geben Sie noch ein Doppelkreuz zur Bestätigung ein. Sicher können Sie sich bereits denken, wie man immer wieder zur Hauptauswahl von BTX zurückkehren kann: Richtig, mit "*0#" gelangen Sie immer wieder zu der Seite, die Sie am Anfang der Verbindung gesehen haben, also zum Hauptmenü von BTX.

Von dieser Seite aus können Sie auch die elektronische Telefonauskunft erreichen, indem Sie einfach eine "81" und dann eine "2" eintippen. Das bringt einige Vorteile gegenüber der normalen Auskunft: Zum einen ist die BTX-Auskunft niemals besetzt (die normale manchmal 45 Minuten lang), erteilt unendlich viele Auskünfte (die normale nur 3, um dann kommentarlos aufzulegen) und zeigt die Adresse des Teilnehmers an (die normale Auskunft darf das nicht). Was passiert, wenn man in München einen guten Bekannten namens Klaus Müller sucht? Das erste Ergebnis der Suche finden Sie in Bild 3.

## Electronic Banking

Ärgern Sie sich auch des öfteren über die merkwürdigen Öffnungszeiten der Banken und Sparkassen? Machen Sie Ihrem Ärger etwas Luft, indem Sie all Ihre Bankgeschäfte über Bildschirmtext tätigen. Hier können Sie Überweisungen

abschicken, Kontoauszüge einsehen, Scheckformulare bestellen - eben all das, was man zum bargeldlosen Zahlungsverkehr benötigt. Bild 4 zeigt Ihnen einen kleinen Überblick über die Möglichkeiten des Electronic Banking. Ein weiterer Vorteil: Die meisten Banken und Sparkassen verlangen für BTX-Geldgeschäfte nur die Hälfte der Gebühren, da sie dadurch weniger Arbeit haben. Am besten erkundigen Sie sich bei Ihrer Bank nach den Vorteilen eines BTX-Kontos.

Natürlich sind nicht nur Bundesanstalten wie die Post oder die Bahn und Banken in Bildschirmtext vorhanden. Sie können beispielsweise auch (eingeschränkt) einkaufen, z.B. können Sie ein Abonnement beim Vogel-Verlag bestellen, sich in Eden ("*EDEN#"), der Stadt der Zukunft, herumtreiben (sofern Sie älter als 18 sind), beim Chaos Computer Club mitmischen ("*CCC#"), Nachrichten an den Reparaturdienst Rat & Tat oder an andere Benutzer des Bildschirmtextsystems schicken oder einfach Informationen wie die aktuellen Charts bei Formel 1 ansehen.

Natürlich kann ich auf dem mir zur Verfügung stehenden Raum das Bildschirmtextsystem nur kurz anreißen, es ist allerdings wesentlich interessanter, als viele BTX-"Gegner" es meistens behaupten. Das Angebot der Post, BTX drei Monate lang kostenlos zu testen, sollten Sie auf jeden Fall in Anspruch nehmen - schaden kann es nicht.

*MP*

*Bild 1*

*Bild 2*

*Bild 3*

*Bild 4*

# NBBS - Multiuser-System mit 16 Ports

**Nachdem in München HOST (das System von Markt & Technik) geschlossen wurde, verlagerte sich ein großer Kreis der HOST-Benutzer auf das NBBS (Nord Bulletin Board System). Da dieses System auf jeden Fall begeistern kann, möchte ich Ihnen eine kurze Vorstellung nicht vorenthalten!**

**Boxname:** NBBS - Nord Bulletin Board System
**Telefonnummer:** (089) 9570081 (8 Leitungen, 300-2400 Baud)
**Datex-Anschluß:** 45 8900 223 (8 Leitungen, 300-9600 Baud)
**Sysopname:** Sysop, Bitwalker
**Online:** 24 Stunden
**Netze:** Fido, weitere in Arbeit
**Steuerung:** Menüsystem, deutsch und englisch
**Schwerpunkte:** Konferenzen, Datenbank, Programme, Bretter
**Download-Angebot:** ca. 10 MByte
**Kosten:** 0,005 DM/Minute

Das NBBS ist ein kostenpflichtiges System. Allerdings wird für die Nutzung kein fester Jahresbeitrag erhoben, sondern es werden zeitabhängige Gebühren abgerechnet, weil auf diese Art und Weise nur dann Kosten anfallen, wenn man tatsächlich eine Leistung in Anspruch nimmt. Alle Kosten werden berechnet nach "Credits", um in kleineren Einheiten als Pfennig rechnen zu können. 2000 Credits entsprechen 1,- DM. Grundsätzlich kann man davon ausgehen, daß eine Minute Online-Zeit bei Nord 10 Credits kostet. Das wiederum läßt sich schwer rechnen und man kann einfacher sagen, daß 10 Minuten 5 Pfennig kosten.

Es wird schwierig, die Möglichkeiten des NBBS' kurz vorzustellen. Ein großer Vorteil ist, daß das Nord Bulletin Board System nicht einen, sondern 16 Anschlüsse hat, die auch von 16 Benutzern gleichzeitig genutzt werden können. Von den 16 Lines sind 8 normale Telefonleitungen (089/9570081) mit 300 bis 2400 Baud, 8 weitere sind Datex-P-Leitungen (458900223) mit 300 bis 9600 Baud. Das System läuft auf einem 33 MHz-AT mit

330 MB-Festplatte und kann problemlos auf 32 Leitungen aufgestockt werden. Ein Zugang über Bildschirmtext ist bereits in Arbeit und steht kurz vor der Vollendung.

## Großes Angebot

Eine herausragende Möglichkeit des NBBS' ist die Möglichkeit des Multiuser-Chats. Das heißt, daß sich (momentan) bis zu 16 Benutzer des Systems gleichzeitig unterhalten können. Die meisten User sind hier zwischen 0 und 2 Uhr anzutreffen, meistens sind mindestens 10 User gleichzeitig da. Ein netter Zug des Systems: In einer "Action List" sind verschiedene Kommandos verzeichnet, die man auf andere Benutzer anwenden kann. Das reicht vom Gähnen über Grummeln zu Winken, Küssen, Treten und Schreien bis zum "Nibblen". Gibt man zum Beispiel "Kiss Sheera" ein, erhält Sheera die Meldung, daß man sie soeben geküßt hat.

Ein anderes hervorstechendes Merkmal ist die Möglichkeit des Compuserve-Outdials. Das heißt, daß man über das NBBS-System auch Compuserve benutzen kann. Compuserve ist das größte Mailbox-Netz der USA. Hierüber kann man beispielsweise alle großen amerikanischen Firmen erreichen. Während der Benutzung des Netzes werden allerdings 280 Credits pro Minute berechnet. Der Compuserve-Account an sich kostet natürlich noch extra Geld, das allerdings über Compuserve selbst abgerechnet wird.

Wie in den meisten Mailboxen existieren auch im NBBS Bretter, in denen Nachrichten geschrieben und gelesen werden können. Weiterhin können knapp 10 Megabyte Public-Domain-Software abgerufen werden, auch für den ST ist viel Software vorhanden.

Den besonderen Flair eines Mailboxsystems machen oft auch die Online-Spiele aus, die man anwählen kann. Mit Kyrandia, dem Multiuser-Online-Adventure des NBBS', scheint ein exzellenter Griff gelungen. Man muß Kyrandia gespielt haben, um es beschreiben zu können - wo sonst kann man schon ein Multiuser-Adventure spielen? Rollenspielfans werden begeistert sein.

*MP*




# PREISAUSSCHREIBEN

## Preise im Wert von über 4000,- DM zu gewinnen

Auch in diesem Monat bieten wir Ihnen wieder ein Preisausschreiben an. Alle, die beim letzten Wettbewerb kein Glück gehabt haben, haben dadurch die Chance, nun doch noch einen der wertvollen Preise zu gewinnen.

In Ausgabe 1 haben wir 5 Interlink-Terminal-Programme von BELA in Eschborn verlost. Die Gewinner stehen nun fest, es sind:

Bernhard Koller, Iserlohn-Oestrich
Wernfred Zolbnhofer, Augsburg
Andreas Kalker, Alsdorf
Rolf Bensch, Wiesbaden
Peter Richter, St. Augustin

Herzlichen Glückwunsch an alle Gewinner! Die Preise gehen den Glücklichen in den nächsten Tagen zu. Für alle, die nichts abbekommen haben, gibt es wegen der recht großen Resonanz ein neues Preisausschreiben. Was gibt es zu gewinnen?

### Btx/Vtx-Manager

2 Btx/Vtx-Manager von Drews EDV + Btx GmbH, Bergheimerstraße 134b, 6900 Heidelberg, Tel. (06221) 29900. Mit dem Btx/Vtx-Manager ist es problemlos möglich, am Bildschirmtextsystem der Deutschen Bundespost teilzunehmen (s. auch Artikel in dieser Ausgabe). Die Btx/Vtx-Manager werden ausgeliefert mit Anschlußmöglichkeit an ein Postmodem, können aber ohne weiteres auch mit einem anderen Modem oder Akustikkoppler betrieben werden.

### MagicBox ST

10 Mailbox-Programme MagicBox ST von Bieling Computersysteme, Spitzwegstraße 11, 4350 Recklinghausen, Tel. (02361) 181485. Mit diesem Mailbox-Programm kann man sich an das MagicNET (siehe Ausgabe 11/89) anschließen und eine eigene Mailbox eröffnen. Dazu fehlt dann nur noch ein Telefonanschluß und ein Modem.

### 3.000.000 Credits

50mal 60.000 Credits im NBBS (Nord Bulletin Board System) der Firma Nord Computer & Software, Emil-Kurz-Straße 1, 8045 Ismaning, Tel. (089) 967572. 60.000 Credits entsprechen etwa drei Monaten freier Online-Zeit im Nord-Computer-Mailbox-System (s. auch Artikel in dieser Ausgabe) und haben einen Wert von 30,- DM.

### CMS-Modem 2400 C

1 CMS 2400 C-Modem (baugleich mit Lightspeed 2400 C) von Compuscience Computerhandels GmbH, Steinwegpassage 5, 2000 Hamburg 36, Tel. (040) 353941. Das Modem beherrscht die Baudraten 300, 1200, 2400 und 1200/75 Baud und ist somit gleichermaßen bestens für Bildschirmtext- und Mailbox-Betrieb geeignet (s. Test in Ausgabe 11/89).

Ein herzlicher Dank geht an die Spender, die uns dieses Preisausschreiben ermöglicht haben.

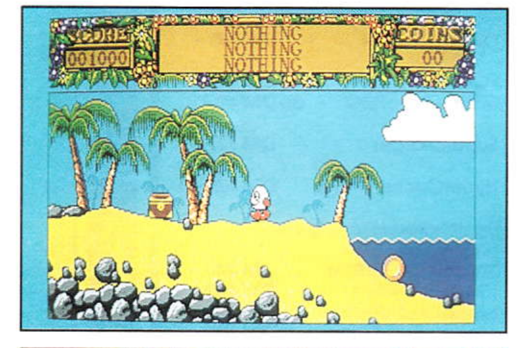
Was Sie tun müssen, um einen der Preise gewinnen zu können? Das ist recht einfach: Schreiben Sie uns einfach eine Postkarte, auf der steht, mit welchem Befehl man grundsätzlich in das Hauptmenü von Bildschirmtext zurückkehren kann (kleiner Tip: Lesen Sie sich den BTX-Artikel in dieser Ausgabe durch). Das ganze schreiben Sie auf eine frankierte Postkarte und schicken sie an

ST-Computer-Redaktion
Stichwort: DFÜ-Preisausschreiben
Industriestraße 26
6236 Eschborn

Bitte vergessen Sie nicht Ihren Absender. Nun bleibt mir eigentlich nur noch, Ihnen viel Glück zu wünschen, um einen der wertvollen Preise zu ergattern. Der Einsendeschluß für das Preisausschreiben ist der 31.3.1990 (Datum des Poststempels), der Rechtsweg ist ausgeschlossen.




Einen putzigen Helden (sieht aus wie ein Frühstücksei mit Armen und Beinen) steuert man in **"Treasure Island Dizzy"** über eine karibische Insel und versucht, von dort zu entkommen - entweder mit 30 eingesammelten Goldmünzen oder ohne. Egal wie, schwierig wird's allemal. Dafür sorgen knackige Rätsel und knifflige Geländeabschnitte, wo ohne Geschick am Joystick gar nichts läuft.






**Treasure Island Dizzy**

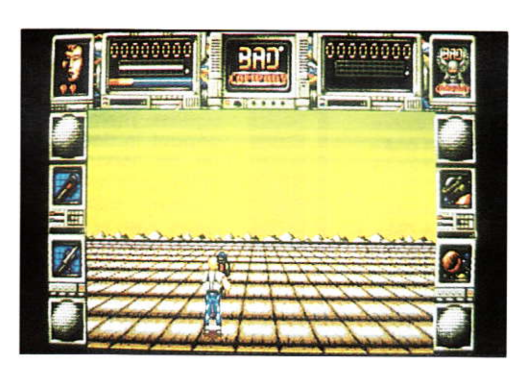





**Maniac Mansion**

In einer verschlafenen Stadt mitten in Amerika verschwindet ein Teenager nach dem anderen. Vermutet wird, daß ein gewisser Doktor Fred in seiner Villa seltsame Experimente mit ihnen treibt. Deshalb machen sich drei Jugendliche auf, um eine Befreiungsaktion einzuleiten. Der Spieler hat Kontrolle über alle drei Helden. Das ist kinderleicht. Um einen Protagonisten auszuwählen, klickt man den gewünschten Namen an. Soll der Gewählte beispielsweise eine Tür öffnen, wählt man in der Verbenliste am unteren Bildschirmrand "open", klickt mit der Maus im Grafikfenster auf die Tür, und wie von Zauberhand läuft der aktivierte Held dorthin und öffnet sie. Humorvolle, spritzige Texte sowie prima Grafiken und die einfache Benutzerführung sind die Qualitäten von **"Maniac Mansion"**. Dieses Adventure sollte in keiner Spielesammlung fehlen.

Zäh wie Leder, hart wie Kruppstahl - auf Mord und Totschlag gedrillt. So sind sie, die durchtrainierten Insekten-Killer aus "**Bad Company**". Auf Tastendruck schickt man gleich ein oder zwei von ihnen in die Schlacht, um vier Planeten zu "kolonialisieren". Schließlich muß die Menschheit irgendwo bleiben, wenn die Welt untergeht. Aufpeitschende Musiken und schöne Grafiken machen die Astro-Metzelei zum Erlebnis.



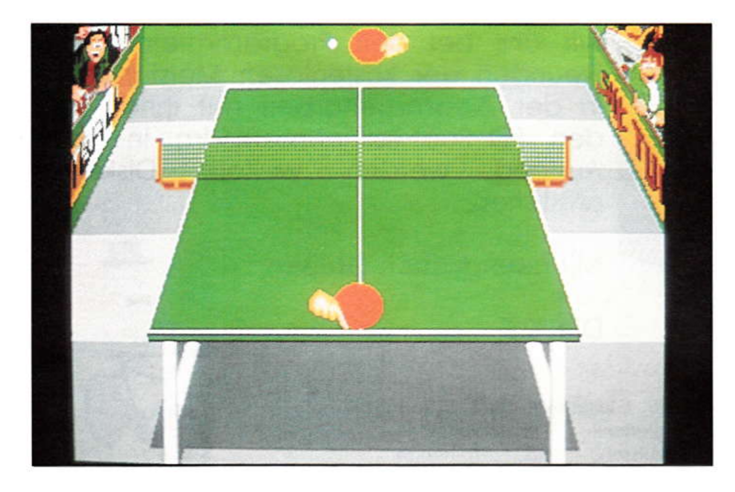



**Bad Company**

Von ARIOLASOFT erscheint in diesen Tagen ein Spiel, das sich deutlich von anderen Spielen abhebt. **Pipe Mania** liegt eine äußerst einfache Spielidee zugrunde. In einem 10x7 großen Spielfeld muß eine Pipeline gelegt werden. Dazu stellt der Computer verschiedene Rohrleitungen zur Verfügung, die aneinanderzulegen sind. Nach einer bestimmten Vorlaufzeit wird der Wasserhahn aufgedreht, und das Wasser bahnt sich den Weg durch die Pipeline. In jedem Level gilt es, eine bestimmte Anzahl von Pipeline-Teilen zusammenhängend zu verbinden. In höheren Leveln gibt es noch diverse Extrapipelines. Ein Zwei-Spieler-Modus ist ebenfalls integriert, wo Teamwork angesagt ist. Eine Demoversion stand schon zur Verfügung. Eines konnte man auch schon nach sehr kurzer Spielzeit feststellen: **Pipe Mania** macht süchtig.

"Who you gonna call? Ghostbusters!" Die Geisterjäger sind wieder unterwegs. Nicht nur im Kino, sondern auch am Computerbildschirm. Im ersten Spielabschnitt wird ein Geisterjäger per Seil in einen Abwasserschacht hinuntergelassen. Ganz unten im Kanal soll er eine Probe magischen Schleims entnehmen. Ist dies vollbracht, eilt man mit der Freiheitsstatue durch New Yorks Straßen und ballert auf angreifende Geisterschwärme, um rechtzeitig in ein Museum zu gelangen. Dort findet der finale Kampf mit Oberbösewicht Vigo statt. Grafik und Sound sind vom Feinsten. Wer auf gut gemachte Action-Spiele steht, wird mit **"Ghostbusters II"** seine wahre Freude haben.

# Table Tennis-Simulator

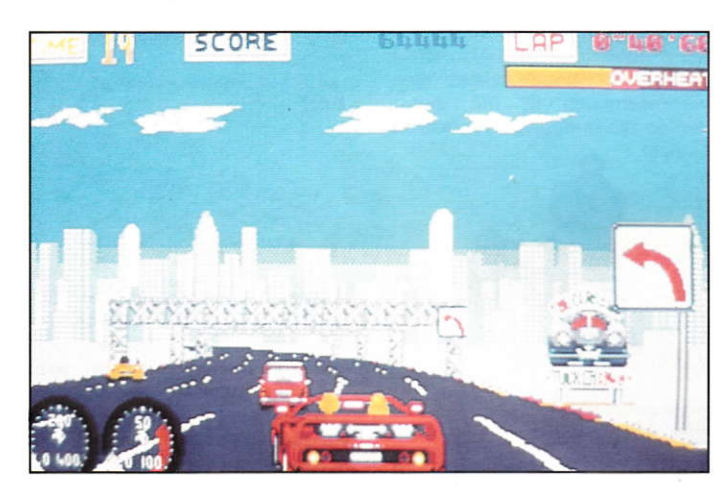


Ein Trainingsmatch gegen die Ballmaschine, Turniere und Einzelwettkämpfe im Einzel oder Doppel, unterschiedliche Schlägerbeläge, Weltrangliste - Starbyte hat beim "Table Tennis Simulator" an alles gedacht, was zum Sport mit der kleinen weißen Plastikkugel gehört. Zu Beginn dudelt der Atari eine erfrischende Titelmelodie, bevor sich der Spieler in den verschiedenen Menüs austoben kann. Beim Trainingsmatch gegen die Ballmaschine kann man noch Tempo, Plazierung und Art des Schlages einstellen, bevor einem das Gerät die Bälle um die Ohren pustet. Nach dem Training und dem richtigen Ballgefühl für die Schläge Slice, Top Spin und den Schmetterschlag wagt man sich in den Wettkampf. Wichtig zu wissen, daß man je nach Wahl des Belages bestimmte Schläge offensiv oder defensiv spielt.Ich beginne mit der Angabe, und der Computergegner schmettert mir die Bälle nur so um die Ohren. Und das irrsinnig schnell. Ich muß wohl noch ein bißchen üben. Nach knapp 30 Minuten verweise ich den ersten Computergegner siegreich von der Platte. Egal, ob ich mit Maus oder Joystick spiele, nach einiger Zeit sehe ich gegen die meisten Gegner gar nicht mehr so schlecht aus. Erstaunlicherweise macht das Hin- und Herschlagen Spaß, wenn auch der Ball oft schwer unter Kontrolle zu bringen ist. Besonders zu zweit gegeneinander oder im Team gegen zwei Computergegner besitzt der Table Tennis Simulator" einen großen Unterhaltungswert. Auch die auf Disc abspeicherbare Weltrangliste tut ihr Übriges für eine relativ hohe Motivation. Nett finde ich auch den Pausengang, bei dem ein Schildkrötenband heiße Rhythmen spielt. Starbytes neues Sportspielchen ist ganz nett gemacht, hat aber keine Hitqualitäten. Da hätten die Bochumer noch mehr an Grafik und Spielbarkeit feilen müssen.

*CBO*




# Turbo Out Run



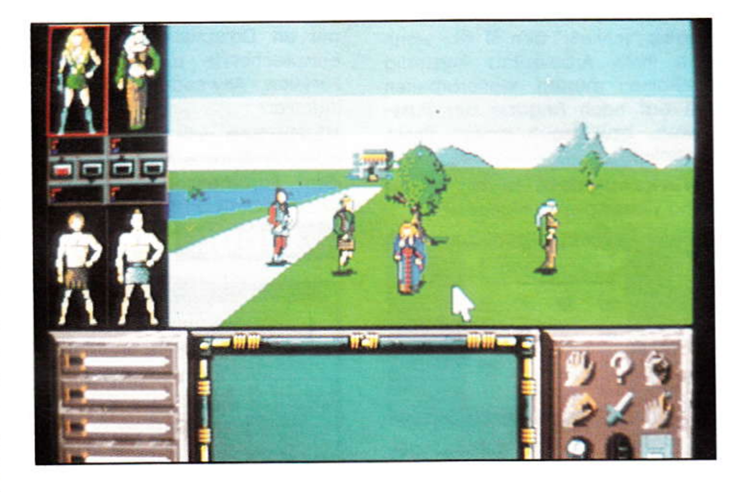

"Out Run" war der größte Renner des Jahres 1988. U.S Gold hat von dem Spiel mit dem schnittigen Ferrari Testarossa weltweit über 750000 Exemplare verkauft. Klar, daß bei soviel Erfolg ein Fortsetzungstitel folgen muß. Und hier ist er: "Turbo Out Run". Bevor das heiße Rennen gegen die Zeit beginnt, wählt der Fahrer zwischen Automatik- oder Schaltgetriebe. Sechs Mechaniker wackeln heran und polieren den Flitzer auf Hochglanz. An der Startlinie drückt der Fahrer seiner blonden Begleiterin noch schnell einen Schmatzer auf die Wange, ein Bikini-Girl hält ein Schild mit der Aufschrift "Ready" in die Höhe, und die Startampel springt von Rot auf Grün. Los geht's! Der Ferrari beschleunigt und rast durch New York.. Wolkenkratzer, Häuser, Bäume und Büsche rauschen, wenn auch nur in ruckeliger 3D-Grafik, vorbei. Wem die Fahrt nicht rasant genug geht, der aktiviert mit dem Feuerknopf den Turbo Booster. Dann sprüht Feuer aus dem Auspuff, und der Flitzer beschleunigt um ein Vielfaches. So düst man über die Highways, überholt Polizeiautos, Porschefahrer, fährt bei Eis und Schnee, bei Tag und bei Nacht durch Städte und Wüsten. Die einzelnen Etappen unterscheiden sich grafisch durch die verschiedenen Hintergründe und Objekte, die am Straßenrand als Hindernisse stehen. Bei hoher Geschwindigkeit aus der Kurve zu fliegen, bedeutet nicht den Tod, aber einen enormen Zeitverlust. Und Zeit ist knapp. Erreicht man die nächste Etappe nicht rechtzeitig, ist das Rennen aber noch nicht vorbei. Fünfmal kann der ambitionierte Ferrari-Fahrer vom Continue-Modus Gebrauch machen, bevor "Game Over" auf dem Screen erscheint. Im Vergleich zum Vorgänger "Out Run" hat "U.S Gold" versucht, das Renngeschehen durch Kleinigkeiten etwas aufzupeppen: Erreicht man beispielsweise einen Checkpoint, spielt eine Musikband am Straßenrand eine Willkommensmelodie. "Turbo Out Run" ist ein gelungenes Autorennen, das genau wie der Vorgänger mit hervorragenden Musiken aufwartet. Auch die Hintergrundgrafiken können sich sehen lassen. Das Ruckeln der Grafiken am Straßenrand ist nicht mehr ganz so schlimm wie beim Vorgängerspiel "Out Run". Es stört kaum noch. Man muß sich schließlich auf die Straße konzentrieren. Grund zur Kritik gibt es allerdings bei den Animationen des Ferraris. Im Vergleich zu Out Run hat der Flitzer viel weniger Animationsstufen, so daß die Unfälle etwas abgehackt aussehen. Wohl nicht mehr genügend Speicherplatz gehabt, was? Doch Fans von "Out Run" werden vermutlich trotzdem mit diesem Spiel glücklich sein.Esist ja auch ganz nett gemacht, aber Hitqualitäten ,wie es englische Kollegen auf der Pakkung bescheinigen, besitzt "Turbo Out Run nicht.

*CBO*



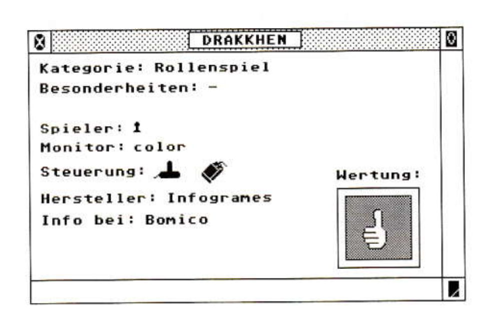
# Drakkhen

Rollenspiele kommen ganz groß in Mode. Immer mehr Spielefans verziehen sich mit Monstern in dunkle Verliese und schmettern Zaubersprüche auf ihre Peiniger. Wenn sich die französische Spieleschmiede Infogrames diesem Thema widmet, sind die Erwartungen sehr hoch gesteckt. Schließlich produziert die Lyoner Company einen Hit nach dem anderen: "Captain Blood", "Hostages", "Kult", "Safari Guns" und "North & South". Um es einmal vorwegzunehmen: Infogrames' neuestes Produkt, das Rollenspiel "Drakkhen", ist wieder ein absoluter Knüller. Zur Einführung in die Handlung liegt der Verpackung ein 78-seitiges Büchlein mit der Hintergrundgeschichte bei. Daraus erfahren wir, daß die riesige Insel, auf der das Abenteuer stattfindet, einst von unheimlich vielen Drachen bewohnt war. Doch fast alle fielen einem furchtbaren Gemetzel zu Opfer. Nur ein alter, weiser Drache hat das Unglück überlebt. Und nur er kann die Insel vor ihrem Untergang retten. Doch der Drachengrufti hat sich irgendwo auf der Insel versteckt. Bevor er eingreifen kann, muß man ihn herbeirufen. Dies geschieht durch den Fund von insgesamt acht Drachensteinen, die irgendwo in acht riesengroßen Dungeons auf der Insel liegen. Die Rollenspielgruppe besteht aus vier Charakteren, die



man zusammen oder einzeln auf der Suche nach den Steinen bewegt. Bei der Steuerung besitzt "Drakkhen" gegenüber anderen Rollenspielen eine außergewöhnliche Neuheit: Drückt man die Return-Taste, laufen die vier Helden zurück, und man sieht die Landschaft aus der Sicht der Gruppe: Jetzt benutzt man Joystick oder Pfeiltasten und bewegt sich in 3D-Vektorgrafik über die Insel. Eine hervorragende Perspektive! Bäume, Sträucher und Gebäude - alles nähert sich so realistisch, als laufe man selber über die Insel. Möchte man die Charaktere wieder einzeln steuern, trifft auf ein Monster oder schickt sich an, ein Gebäude zu betreten, tauchen wieder alle lebendigen Mitglieder der Gruppe auf, und man hat Kontrolle über jeden einzelnen. Auf der Suche nach den acht Drachensteinen trifft die Gesellschaft auf alleinstehende Häuser, Städte, Läden, Dungeons und natürlich auf einen Haufen Monster. Alle präsentieren sich in hübsch animierter Grafik in unterschiedlichsten Formen. Es gibt riesige Saurier, dicke Würmer, muskelprotzige Rotzwerge und vieles mehr. Die Monster lassen sich durch Schwerterkampf von einem einzelnen Charakter oder der ganzen Party vernichten. Natürlich stehen auch eine Vielzahl von Zaubersprüchen zur Verfügung, die wie Hieroglyphen aussehen. Richtig benutzen kann man sie nur, wenn man über das mitgelieferte Handbuch verfügt (ein wirksamer Kopierschutz!). Die anderen Aktionen lassen sich in einem Menü mit grafischen Symbolen einleiten. Man kann Gegenstände untersuchen, sie aufheben, mit Faust oder Schwert kämpfen, geheime Mechanismen bzw. Fallen auslösen und einiges mehr. Gegenüber anderen Rollenspielen bringt "Drakkhen" kaum spielerische Neuheiten. Dafür setzt "Drakkhen" technische Maßstäbe im Rollenspielgenre - sowohl in puncto Grafik als auch beim Sound, der alle Situationen passend untermalt. Von Infogrames neuestem Werk werden Fans von Magie und Monstern noch lange schwärmen.

*CBO*



DRAKKHEN

Kategorie: Rollenspiel
Besonderheiten: -

Spieler: 1
Monitor: color
Steuerung:
Hersteller: Infogrames
Info bei: Bomico

Wertung:

# Rings of Medusa

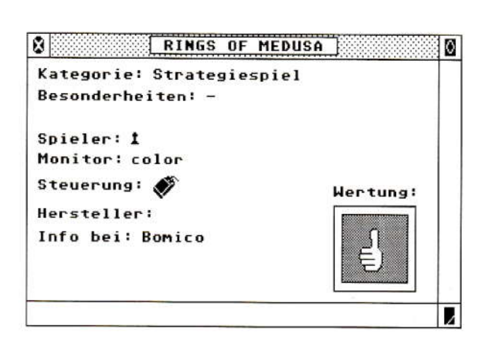
Jeden Monat werfen die Software-Häuser Hunderte von Spielen auf den Markt. Jedes Programm findet seine Käufer. Aber nur wenige entwickeln sich zu absoluten Hits. Einer dieser Megaverkaufsrenner ist "Rings of Medusa". Noch bevor "Starbyte" das Spiel fertig programmiert hatte, lagen bei der Vertriebsfirma Bomico bereits 5000 Vorbestellungen vor. Kein Wunder. "Rings of Medusa" ist gigantisch gut. Adventure, Strategie- und Handelssimulation sind komplex miteinander verknüpft. Das garantiert wochenlangen Spielspaß allererster Güte. Bei "Rings of Medusa" handelt es sich um ein Strategiespiel, vergleichbar mit "Fugger" oder "Hanse", nur um einige Klassen besser. Das liegt an der für Strategiespiele außergewöhnlich gelungenen Grafik, den mannigfaltigen Spielmöglichkeiten und



der spannenden Handlung: Die Hintergrundgeschichte versetzt den Spieler in ein Königreich, wo alle in Friede, Freude und vor allem Eierkuchen leben. Bis eines bösen Tages die hinterlistige Göttin Medusa versucht, die Menschen zu unterjochen und das einst so friedliche Königreich in ein Land voller Kriege und Unruhen zu verwandeln. Den finsteren Machenschaften Medusas muß ein Ende bereitet werden. Deshalb macht sich der wagemutige Königssohn auf den Weg, um der Göttin in einem Zweikampf den Garaus zu machen. Zum alles erlösenden Duell kommt es allerdings erst, wenn der Held im Königreich fünf magische Ringe gefunden hat. Klingt so, als sei "Rings of Medusa" ein klassisches Adventure. Ist es aber nicht. Im Vordergrund steht das Strategiespiel, aber auch die Handelssimulation. Um das abenteuerliche Spielziel zu erreichen, sprich die fünf Ringe zu finden, braucht man ständig Geld für eine schlagkräftige Armee, ohne die man die beschwerliche Suche nach den Ringen auf die Dauer nicht überlebt. So ist man also ständig damit beschäftigt Geld herbeizuschaffen. Da gibt es verschiedene Möglichkeiten: zum

Beispiel als Schatzsucher. In den Tempeln der Städte befinden sich Koordinaten für geheime Verstekke. Sucht man in dem angegebenen Gebiet, findet man Rohstoffvorkommen, Geldschätze und manchmal auch einen der magischen Ringe. Gemeinerweise muß man viele Städte erst erobern, bevor der ATARI die Koordinaten verrät. Sehr gut Geld verdienen läßt sich mit dem Handel. Man kauft in irgendeiner Stadt irgendeine Ware günstig ein und verscherbelt sie in einer anderen teurer. Mit dem Profit läßt sich die Armee oft entscheidend vergrößern. Risikofreudige Zeitgenossen wagen sich ins Spielcasino und versuchen, ihr Geld beim Black Jack zu gewinnen. Andere gehen zur Bank und lassen sich einen Kredit geben, den sie allerdings auf Heller und Pfennig mit sehr hohen Zinsen zurückzahlen müssen. "Money makes the world go round". Das gilt auch bei "Rings of Medusa". Um effektiven Handel zu betreiben, bedarf es Fuhrwerken und Pferden, die es in Geschäften teuer zu kaufen gibt. Manche Ringe befinden sich auf entfernten Inseln im Ozean. Um dorthin zu gelangen, braucht der Spieler Schiffe und natürlich Matrosen, für die genau wie für alle Armeesoldaten ständig Soldzahlungen einkalkuliert werden müssen. Ja, Armee und Matrosen sind teuer. Deshalb schmerzt es gewaltig, wenn Soldaten in Gefechten mit feindlichen Streitkräften oder Räubern fallen. Bei solchen militärischen Auseinandersetzungen darf man keine schillernden Schlachten auf dem Bildschirm erwarten. Die Stärke der kämpfenden Truppen wird gegeneinander aufgerechnet, Trainingsstärke und Zufallsgenerator spielen noch ein bißchen hinzu, und das Ergebnis der Schlacht wird auf dem Bildschirm ausgegeben. Die Effektivität der Armee hängt im wesentlichen davon ab, welche Leute man auf die einzelnen Posten setzt. Im Königreich gibt es viele Rassen, angefangen von menschlichen Soldaten bis hin zu Zwergen und dicken Trollen. Der eine schlägt sich hervorragend bei der Kavallerie, der andere ist besser bei der Infanterie aufgehoben. Welche Rasse auf welchem Posten kämpft, muß der Spieler ebenfalls herausfinden. Ich könnte jetzt noch mehrere Seiten über die vielen spielerischen Möglichkeiten von "Rings of Medusa" schreiben. Aber das würde den Rahmen dieses Artikels sprengen. Nur soviel sei gesagt: Starbytes neues Strategiespiel ist so komplex wie kein anderes! Trotz dieser Komplexität wartet das Programm mit hervorragender detailreicher Grafik auf. Und das Schöne ist, man braucht keine Zahlen oder Befehle einzutippen oder dem Held Richtungsangaben zu machen. Alles ist komplett mausgesteuert. Selbst den Königssohn steuert man per Maus durch die vogelperspektivische Abenteuerwelt. Alle anderen Aktionen werden mit Hilfe von Symbolen am unteren Bildschirmrand eingeleitet. "Rings of Medusa" ist ein Erlebnis. Idee und Motivation verdienen die Höchstnote. Wer sich das Spiel zulegt, für den wird Langeweile ein Fremdwort. Empfehlenswert ist "Rings of Medusa" allerdings nur für fortgeschrittene Strategiespieler. Für blutige Anfänger dürfte das knifflige Game ein wenig zu schwierig sein.

*CBO*



# In der nächsten ST-Computer lesen Sie unter anderem

## *Riemann*

Riemann ist ein Programm für symbolische Algebra und Analysis. Ist Riemann für Mathematiker, Schüler und/oder Studenten interessant? Kann man komfortabel mit dem Programm arbeiten? Wird ein Coprozessor unterstützt? Wenn Sie Antwort auf die Fragen suchen, sollten Sie sich die nächste Ausgabe der ST-Computer zulegen.

## *150 MB-Streamer*

Wer viele Daten hin- und herschiebt, braucht oft auch Sicherheitskopien. Was liegt näher, als einen Streamer anzuschließen, der alle wichtigen Daten in einem Rutsch sichert? Wie schnell der neue Weide-Streamer ist, wieviel er wirklich speichert, ob man ihm unbesorgt seine Daten anvertrauen kann, erfahren Sie in der April-Ausgabe der ST-Computer.

## *Scanface ST*

Bisher gab es diverse Scanner für den ATARI ST, die immer fast immer eine spezielle Anpassung benötigten. Bei Scanface ST ist man einen anderen Weg gegangen und hat ein universelles Interface für alle drei Panasonic-Scanner gebaut, das einfach extern zwischen ST und Scanner geschaltet wird. Was das Interface und die mitgelieferte Sofware leisten, lesen Sie demnächst in der ST Computer.

**Die nächste ST Computer erscheint am Fr., den 30.3.90**

## Fragen an die Redaktion

Ein Magazin wie die ST-Computer zu erstellen, kostet sehr viel Zeit und Mühe. Da wir ja weiterhin vorhaben, die Qualität zu steigern, haben wir Redakteure ein großes Anliegen an Sie, liebe Leserinnen und Leser:

Bitte haben Sie Verständnis dafür, daß Fragen an die Redaktion nur **Donnerstags von 14[00]-17[00] Uhr** unter der Rufnummer 06196/481814 telefonisch beantwortet werden können.

Natürlich können wir Ihnen **keine** spezielle Einkauftips geben. Wenden Sie sich in diesem Fall bitte an einen Fachhändler. Wir können nur Fragen zur ST Computer beantworten.

*Vielen Dank für Ihr Verständnis*


## Impressum ST Computer

**Chefredakteur:** Uwe Bärtels (UB)
**Stellvertreter:** Harald Egel (HE)

**Redaktion:**
Uwe Bärtels (UB)
Harald Egel (HE)
Harald Schneider (HS)
Martin Pittelkow (MP)

**Redaktionelle Mitarbeiter:**
C.Borgmeier (CBO)
Claus Brod (CB)
Ingo Brümmer (IB)
Derek dela Fuente (ddF)
Stefan Höhn (SH)
Raymund Hofmann (RH)
Dieter Kühner (DK)
Claus P. Lippert (CPL)
Markus Nerding (MN)
Chr. Schormann (CS)
R.Tolksdorf (RT)
Thomas Werner (TW)

**Autoren dieser Ausgabe:**
R.Blittkowsky
J.Bolt
H.Büche
P.Denk
M.Fritze
H.Illsinger
K.D.Litteck
M.Macher
C.Schmitz-Moormann (CSM)
P.Neuchel
R.Peiler (RP)
D.Rabich
S.Schäfer
U.Seimet
R.Storn
R.Wiechert

**Auslandskorrespondenz:**
C.P.Lippert (Leitung), D.Dela Fuente (UK)

**Redaktion: MAXON Computer GmbH**
Postfach 59 69
Industriestr. 26
6236 Eschborn
Tel.: 0 61 96/48 18 14, FAX : 0 61 96/4 11 37

**Verlag: Heim Fachverlag**
Heidelberger Landstr. 194
6100 Darmstadt 13
Tel.: 0 61 51/5 60 57, FAX : 0 61 51/5 56 89 + 5 60 59

**Verlagsleitung:**
H.J.Heim

**Anzeigenverkaufsleitung:**
U.Heim

**Anzeigenverkauf:**
K.Margaritis

**Anzeigenpreise:**
nach Preisliste Nr.3, gültig ab 1.1.88
ISSN 0932-0385

**Layout:**
Kerstin Feist, Martin Lowack, Manfred Zimmermann

**Titelgestaltung:**
Gunter Wenzel (Tel.: 06172/37193)

**Fotografie:**
Martin Lowack, Klaus Ohlenschläger

**Produktion:**
K.H.Hoffmann, B.Kissner

**Druck:**
Frotscher Druck GmbH

**Lektorat:**
V.Pfeiffer

**Bezugsmöglichkeiten:**
ATARI-Fachhandel, Zeitschriftenhandel, Kauf- und Warenhäuser oder direkt beim Verlag

**ST Computer erscheint 11 x im Jahr**
Einzelpreis: DM 7,-, ÖS 56,-, SFr 7,-
Jahresabonnement: DM 70,-
Europ. Ausland : DM 90,- Luftpost : DM 120,-
In den Preisen sind die gesetzliche MWSt. und die Zustellgebühren enthalten.

**Manuskripteinsendungen:**
Programmlistings, Bauanleitungen und Manuskripte werden von der Redaktion gerne angenommen. Sie müssen frei von Rechten Dritter sein. Mit seiner Einsendung gibt der Verfasser die Zustimmung zum Abdruck und der Vervielfältigung auf Datenträgern der MAXON Computer GmbH. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte wird keine Haftung übernommen.


COMPUTERVIREN
Das neue
Anti Viren Kit 3.0
von G DATA
Jetzt auch für MS DOS*
und kompatible Computer:
ANTI VIREN KIT PC: DM 198,–**
Version für PC Speed: DM 99,–**
* MS DOS ist ein eingetragenes
Warenzeichen der Microsoft Corp.
** Unverbindliche Preisempf.
Kostenloser
Update-Service
für registrierte Benutzer.
Mit Harddisk-Überwachung.
Diagnose von Link-Viren.
Das meistverkaufte
Anti-Viren-Programm
für ATARI ST Computer
Anti Viren Kit 3.0
Anti Viren
DM 99,–
(unverbindl.
Preisempfehlung)
PROGRAMMABSTURZ!
DATENVERLUST!
Neue Version mit
Harddisk-Schreibüberwachung
und Virenscannung
Anti Viren Kit 3.0
bei Ihrem Atari-Händler
oder direkt bei
G DATA
G DATA
Bundesrepublik:
Siemensstraße 16, D-4630 Bochum 1
Telefon. Bestellungen: 0 23 23/38 98 58
Versand zzgl. DM 5,– Portopauschale
per Nachnahme oder Vorausscheck.
Schweiz:
Computertrend, Langstr. 31, CH-8021 Zürich
Österreich:
CSR Computer Rittner, Hauptstraße 34,
A-7000 Eisenstadt




ST-Computer März 90